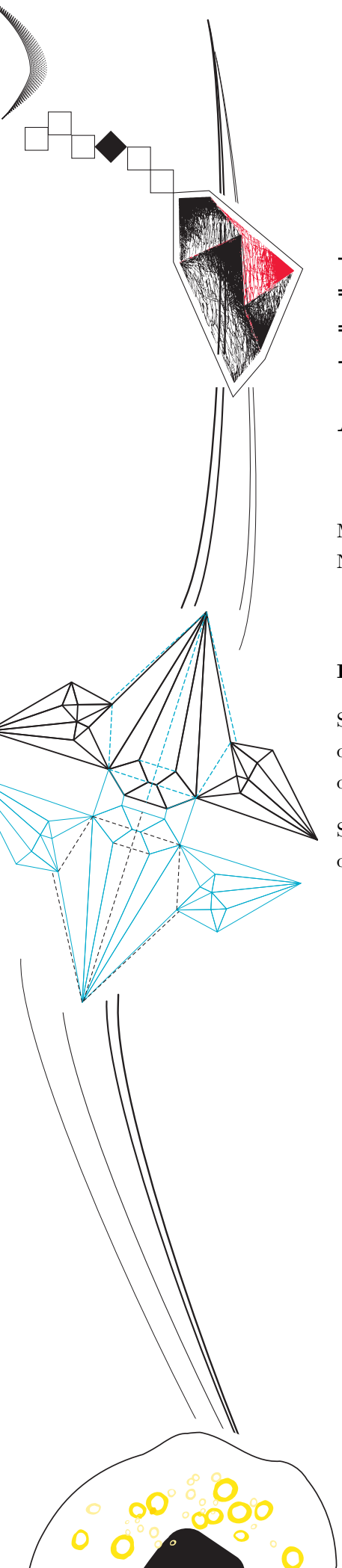

November 2019

IMPROVING VEHICLE ROUTING BY UTILIZING REALISATION DATA

A CASE STUDY

Master thesis by:
N.P.M. Clerkx

Examination Committee

Supervisors University of Twente:
dr. ir. M.R.K. Mes
dr. ir. E.A. Lalla

Supervisor ORTEC:
dr. ir. P. van 't Hof

Management Summary

In an era where the availability of data is ever growing, opportunities arise for improving vehicle routing software through data analytics. Where Operational Research focusses on solving a predefined problem, data analytics can help in improving the definition of this problem. In this research, we develop a methodology that uses the data that is gathered during execution to improve the performance of vehicle routing software by adjusting the handling time parameters. A case study is performed at a distribution centre of a large retail client of ORTEC.

Currently, the handling times in ORTEC's vehicle routing software are configured based on estimations by process experts. No analysis on the effects of altering the handling times is performed. Changing handling times affects costs made in the planning and costs made by missing delivery windows during execution. A trade-off occurs between these two types of costs. If we increase handling times, the costs of the planning will increase since the total duration increases. However, the number of missed deliveries will decrease because of the increased amount of extra time that is planned for the deliveries.

To find the best handling time configuration, a method is developed that iteratively evaluates handling time configurations by planning and simulating new configurations. The developed methodology consists of four parts: data extraction and cleaning, data exploration and distribution fitting, simulation model building and validation, simulation optimization. The input for this methodology is a year's worth of planning and realisation data from the aforementioned retail client.

The planning and realisation data was extracted from the database using a query that joined the data on the coarsest level of data granularity. Inspecting and visualizing the resulting data set proved to be important to find erroneous data. To detect and remove statistical outliers, the double Mean Absolute Deviation method was applied, as this can deal with non-normal skewed distributions.

Three sets of distributions for different sources of randomness have been fitted on the data. These sources are the stop durations, travel durations and start time deviation. The selection of features that should be accounted for in the distributions was done based on insights gained from correlation between variables and the feature importance as calculated by an Random Forest (RF). For the stop and travel durations a simple statistical approach was compared to Random Forests for Conditional Density Estimation (RFCDE) on the cross-validated Average Conditional Log Likelihood (ACLL) to determine the best fitting model. In both cases, the RFCDE proved to be the better performing model.

A simulation model was created by combining a list of assumptions with the distribution models. The simulation model was validated by comparing historic realisations with simulated

realisations that were created using the same historic planning.

Optimizing the handling times was assisted by a reparametrization model that converts a fraction of stops to be completed within their planned time to a set of handling time settings. To iteratively find the best handling time percentile, a Gaussian Process (GP) was used to estimate the relation between the input and output of the simulation and the Expected Improvement policy was used to determine new inputs to measure. Using an estimated weight to combine the costs of the planning with costs made by missing deliveries, we have shown that our optimization algorithm performs well in a noisy measurement environment. The optimization algorithm was able to find the location of the estimated minimum in 15 iterations with an error of 3.5% compared to the location of the estimated minimum after 30 iterations.

A method for determining input setting regions of interest was shown, for when the weights with regard to the plan costs and the time window violations are unknown, as was the case in our research.

Finally, a proposed input setting was shown to have 0.12% lower plan costs and 3.73% less time window violations as compared to the retail client's current setting. This might be improved further by looking into adding more detailed settings.

From this all we can conclude that there is value to be gained from the usage of data analytics in combination with vehicle routing. We therefore recommend that this methodology is further productised at ORTEC.

Preface

This thesis marks the end of a project I enjoyed working on. In this project, I was able to combine several very interesting subjects I encountered during my master Industrial Engineering and Management at the University of Twente: Operations Research, Data Science and programming. As I would not have been able to complete this project without the help of my supervisors, co-workers at ORTEC and my friends and family, I would like to take this opportunity to express my gratitude.

I would like to thank Martijn for the excellent guidance and challenging me during the project. Encouraging me to not only evaluate but also optimize has increased the quality of this thesis profoundly. I also want to thank Eduardo for taking the time to provide thorough and constructive feedback to my, sometimes short, reports. This has definitely helped me improve the structure and theoretical relevance.

Although there are many people at ORTEC who have helped me in the last eight months, I would like to especially thank Pim. Your support and encouragement have been very valuable and I will always remember our trip to Moscow that marked the start of this project! I would also like to thank Leendert and Taco for their ideas and insights.

I would also like to thank my family as, although I have not been home quite as much as I would have liked, I could always count on their support. I would like to thank all my friends for making my student time a period in which, besides studying, had a lot of fun. Finally, I would like to thank my girlfriend for her everlasting support and encouragement.

Nathan

Contents

Abbreviations

2L-MCVRP Two-dimensional Loading and Multi-Compartment Vehicle Routing Problem

ACLL Average Conditional Log Likelihood

BO Bayesian Optimization

CDE Conditional Density Estimation

CDF Cumulative Distribution Function

DC Distribution Center

GP Gaussian Process

HVRP Heterogeneous Vehicle Routing Problem

KDE Kernel Density Estimation

KPI Key Performance Indicator

MILP Mixed Integer Linear Program

NN Neural Network

OR Operations Research

ORD ORTEC Routing and Dispatch

QRF Quantile Regression Forest

RF Random Forest

RFCDE Random Forests for Conditional Density Estimation

SDVRP Split Delivery Vehicle Routing Problem

SVRP Stochastic Vehicle Routing Problem

SVRPTW Stochastic Vehicle Routing Problem with Time Windows

TSP Travelling Salesman Problem

VRP Vehicle Routing Problem

VRPTW Vehicle Routing Problem with Time Windows

Chapter 1

Introduction

This research is performed at ORTEC, which is a company that is specialized in making software products that aid their customers in creating more value by optimizing existing business processes through software and mathematics. ORTEC's optimization software can handle many types of problems, including warehouse control, pallet and space loading, workforce scheduling and fleet routing and dispatch. The software product this research focuses on is the vehicle routing software called: ORTEC Routing and Dispatch (ORD). This software is typically used by a wide variety of companies to plan routes from depots and deliver orders to clients or establishments.

ORD ensures that not only the individual routes are feasible but also the overall schedule can be executed. In order to make ORD provide the companies with schedules that are realistic and comply with all constraints and wishes, the software is made highly configurable. Due to the large problem size and the numerous restrictions, it is not possible to incorporate stochastic information on input data into the optimization process. Therefore some input data of ORD are deterministic approximations of real-life stochastic variables. Input data values corresponding with stochastic variables in the real system mainly concern durations of actions that have to be planned, like (un)loading or travels. Currently, the deterministic approximations of these durations are estimated by process experts based on experience.

The goal of this thesis is to provide data-driven methods for setting these input data values, such that planning and the accompanying execution improves. How this improvement is measured will be discussed further in Section 1.2. The data that is used is gathered during the execution of the routes. For the remainder of this thesis, when we talk about input data values, only the input data values that are stochastic in real-life are meant. Other data that can be seen as input for the planning software, like problem instances or constraints, are dependent on the requirements of the user of the planning software. Since these requirements are often fixed and cannot be altered to increase performance of the software, these are not considered to be tunable input data in this thesis.

This chapter will further introduce this research. Section 1.1 gives some motivation as to why this research is conducted. Section 1.2 gives an identification of the problem this research aims to solve. In Section 1.3 the objective and research questions are given and Section 1.4 defines the limitations of this research.

1.1 Motivation

Improving the schedules generated by ORD (or any other vehicle routing software package) can be done by either adjusting the algorithmic parameters or the input data. The former is quite extensively addressed in literature and at ORTEC. Accounting for the stochasticity of input data during optimization of the routes is currently infeasible for large instances due to computational resource limits. Adjusting the input data is something that has not been much researched. The lack of a structured method to continuously improve the input data makes the systems in place dependent on the estimation of real world durations by users or experts. Changes in the real world values of the input data over time may go unnoticed by these experts, leaving the optimizing software with a broken or outdated view on the situations it has to optimize. Also dependencies of input data values (e.g., certain vehicles have longer handling times) are hard to discover and quantify by hand without statistical evidence. Developing a methodology to systematically adjust and verify these adaptations of the input parameters is useful for all users and developers of routing optimization software.

ORTEC has a strong background in the Operations Research (OR) field and has acquired some knowledge on data science over the years. Currently some data science techniques are already used in dedicated studies, as teaching material and in some forecasting engines integrated into software products. Further incorporating data science techniques into optimization products is a strategic goal for ORTEC. This research aims to describe a methodology for improving the input data values so that ORTEC can apply this to the software installed at multiple customers.

1.2 Problem identification

Since the current process for setting the input data values is simply taking an educated guess, the required contribution of this research is twofold. First, the stochastic information on input data values has to be obtained from a dataset. Second, from this stochastic information the best deterministic values have to be chosen. If only the first step would be done, we could simply select all mean data values and use these in future planning. However, using the mean as setting for a duration will cause roughly half of the actual durations to be longer than planned and the other half be shorter than planned. As there may be costs or service level agreements connected to arriving late at a location, configuring the mean will most likely not be the best setting. Therefore, the second step also has to be performed to find the best setting. To figure out what the best setting is, we have to be able to measure how good or bad a setting is. When algorithmic parameters are tuned, the assumption is that the input data is correct. This means that an improvement in performance for an alternative set of algorithmic parameters, can be measured by looking at the Key Performance Indicators (KPI) of the planning. When input data values are tuned, this is not possible. For instance, when we underestimate the time it takes to unload pallets, the KPIs of the planning would only improve because the total time would decrease. In reality however, the planning would cause the driver to potentially miss the next time window since he can never unload the pallets that quickly. This means that this research will have to develop a method that is able to deal with KPIs related to route *realizations*. This also makes the method previously developed by Demkes (2014) to tune algorithmic parameters of

ORD unsuitable for tuning the input data, since this method only uses the KPIs of the planning to tune the software. Since our learning method will have to deal with KPIs of both the planning and of the realizations, a trade-off will occur between these.

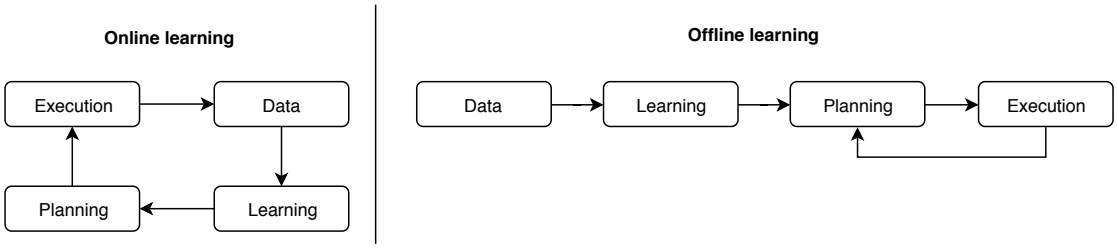

Figure 1.1: Schematic of on- and off-line learning, left and right respectively

Learning methods can be divided into two categories: online and offline. In Figure 1.1, the difference can be seen. When applied to the situation at hand, online learning would include the learning step into the daily process, updating the input data before creating a planning each day. At the end of the day the observations would be used to determine how the input data should be altered to improve the performance. This is in contrast with offline learning, where a set of data is used once to adjust the parameters and simply use these afterwards. For some practical reasons, online learning is not suitable for this research. First and foremost is the fact that an online learning method would have to explore the solution space in order to find the input data settings that perform well. That knowledge then has to be exploited in order to converge to the best setting. This would make the method experiment with settings in the real world that may give bad plannings, and hence are expensive. From a business perspective this is undesirable for both, ORTEC and the client. Also, the scarcity of the computational resources during daily usage of the software contributes to the unsuitability of an online learning method. This means the solution of this research must be sought in the corner of the offline learning methods.

An offline method only has a set of historic data to learn from. How this learning will take place is not straightforward however. Changing the input settings will change the planning the software will generate. The performance of the adjusted input settings can only be measured if a planning and the accompanying realisation is known. Therefore, since our learning method will be offline and no data is present for the new plannings, the learning method has to be able to simulate realizations that go with this new planning. Only then can the performance of the proposed input setting be estimated. The schematics of this learning process can be seen in Figure 1.2. In order to simulate the realised durations of a schedule, it is necessary to estimate the distributions of these stochastic values. A method is needed to determine on what variables (e.g., location, truck or day of the week) the distribution of the durations are dependent.

The problem that then remains is selecting the values of input settings to test. A problem that was encountered by Demkes (2014) was the large amount of time it takes for the software to create a planning. When a large computational time is needed for each run, it is not feasible to simply run all possible options. This was solved by using a form of online algorithm that sequentially chooses the next input setting to test. Where Demkes (2014) only needed to test a configuration by running the planning software, the method in this research will also have to

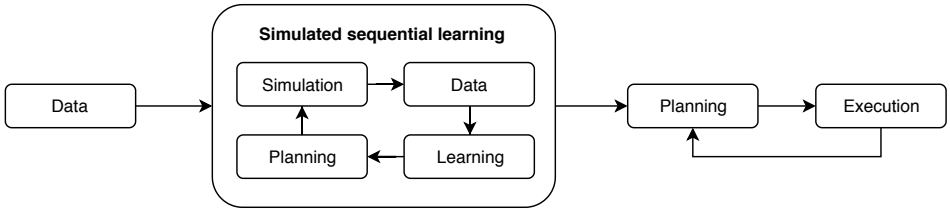

Figure 1.2: Schematic of offline learning with an sequential learning step

incorporate a simulation. So in each run the learning algorithm will have to choose an input data setting to test, create a planning and run a simulation of that planning. When the KPIs of the simulation are observed, this knowledge can be used to determine the next setting to test.

The long computation time of a single run and the importance of the KPIs of the realizations result in the need for an offline method with a simulated online component. The schematics of the proposed type of method can be seen in Figure 1.2. As shown in the figure, the proposed method intends to use the historic data once to create a simulated environment. In this simulated environment, a learning method will sequentially chose which setting to evaluate. This sequential learning method will eventually converge to the best input data setting. The output of the learning method is a suggested improved input data configuration that then can be used in daily operations.

The method that will be developed will be applied at a single customer of ORTEC that uses ORD. This customer will be referred to as the Retailer. The routing case of the Retailer and the data that will be used is explained in Chapter 2.

1.3 Objective and research questions

In Section 1.1, the need for a learning method is explained, and Section 1.2 argued what form it should have. From this we can formulate the goal of this research:

> *Develop an offline learning method that finds better input settings to improve plannings made by ORD*

To achieve this objective, we define multiple research questions where some have sub-questions. The research questions and the reasoning behind these questions are given below.

To effectively solve the problem we first need some information on this problem, information will be gathered on the routing case present at the Retailer. Also the current configuration and tunable input parameters are investigated. The data that will be used in the remainder of the research will be explained and briefly examined.

1. In what context is this research conducted?
    - Which input settings can be adjusted in ORD and how are these currently configured at the Retailer?
    - What are all characteristics of the routing case?

- What KPIs are used to measure the performance of a schedule?
- How can we quantify KPIs related to the realisation?
- What data is available?

After we have gathered information on the specific problem that will be solved in this thesis, we need information on how we could do this. This information will be gathered from current literature on the relevant subjects. To see how the quality of plannings can be estimated, we will look into the vehicle routing literature for a comparable version of the problem that is present at the Retailer. Also previous research that has been done on tuning input data in routing problems will be investigated. After that, techniques that can be used for setting up and optimizing a simulation model are looked into.

2. What literature is available related to stochastic vehicle routing, data mining, simulation and sequential learning?

    - What type of the Vehicle Routing Problem (VRP) is related to our case and how are these typically solved?
    - What methods are available to infer distributions from data?
    - What methods are known for setting up and validating a simulation model?
    - What sequential simulation optimization methods are available?

After obtaining the knowledge from literature on how this type of problem is handled and what techniques can be used for this, these techniques will be applied to create a simulation model for the routing case of the Retailer. To achieve this, the data gathered at the Retailer will be used to create and validate the model.

3. How can we use the realisation data to perform a simulation?

    - What numbers have to be simulated?
    - How can we determine from the data what factors influence the variables that have to be simulated?
    - What method is most suited to estimate the distributions?
    - What assumptions have to be made?
    - How can we validate our simulation model?

When a simulation model has been created that is valid, the near optimal setting has to be found. For this, a suitable sequential simulation optimization algorithm is needed. Since optimizing over a large number of parameters can be a difficult problem, we want to use all knowledge on the problem that is available to help the optimization process. We would like to know how well the algorithm performs, and what the impact is of the new settings on the routing case of the Retailer.

4. How can a sequential learning algorithm be used to find the best settings?

    - How can the available data be used in the optimization process?

- How well does the algorithm perform?
- How fast does the tuning method find a good set of input settings?
- How well do the new input settings perform compared to the old input settings?

The method that is developed in this thesis is applied to a single customer case. Therefore, the adjustments that have to be done to implement the methodology so that it can be used for other customer cases will be discussed.

5. How can ORTEC implement the methodology so the learning method can be used for other cases?

To meet the research goal, the research questions will be answered in the order they are stated. This thesis is organised such that each chapter answers one of the research questions. The first question will be answered by talking to content experts within ORTEC and at the Retailer. The second question will require a literature study to answer. The following questions concern the development and execution of the method. In the final chapter, conclusions and recommendations will be made regarding all previous chapters.

1.4 Limitations

To manage the size of this research, some limitations are defined to clarify what will be investigated in this thesis. The limitations that could be encountered during the research can be found below.

- **Input data corresponding to stochastic random variables**
  Only input data corresponding to variables of a stochastic nature in the real system are investigated. No changes will be made to either algorithmic parameters or constraints.
- **Current configuration**
  Only the values that are currently configured will be tuned. No new input data settings will be added.
- **Data quality**
  The quality and availability of the data is important for the success of developed method. When issues arise during the research because of this, some assumptions may have to be made or data has to be generated to be able to complete the method.

Chapter 2

Context

This chapter will explain and discuss the context in which this research is conducted. Section 2.1 will discuss the software ORD and the configuration options it has. In Section 2.2 the Retail case and the available data is described.

2.1 Planning software

This section will first give a short explanation of the routing terminology of ORD that is used in the remainder of this thesis. The different options for configuring ORD are described in Section 2.1.2.

2.1.1 Terminology

There is a lot of technical terminology in this thesis. A lot of terminology is related to VRP, but since researchers and practitioners often have different explanations for the same words, the meaning can be ambiguous. Therefore it is useful to give an overview of how the terms are used at ORTEC. These terms will also be used in this thesis. The terms in this thesis are defined as follows:

Order: An amount of a single product that needs to be delivered from a depot to a customer. For instance, 3 pallets of frozen goods.
Resource: An entity required to perform certain actions. Most common resources are: trucks, trailers and drivers. Although present, this thesis does not deal with trailers and drivers.
Action: An activity in a trip. Examples are a stop action and a travel action.
Trip: A sequence of actions performed by a resource starting and ending at a depot. In literature often referred to as routes.
Depot: A central warehouse where the orders are present prior to delivery.
Customer: A location where orders are delivered.
Stop action: The period a truck is at a single location. This can be at either a depot or a customer. During a stop orders are picked up or delivered, these are then called pickup- or delivery-stops. The planned duration of a stop is location and amount dependent.
Travel action: Section of a trip where the truck travels between two locations. Also referred

to as sections, edges or legs.

Address opening time: Time period in which the location is open such that (un)loading can take place.

Time window: Time period in which an order has to be delivered at a customer. Differs from address opening times in that a user of ORD can set these separately for all orders. These are usually more restrictive than the address opening times.

Address type: A grouping of addresses based on some common attribute that is configured by the user of ORD. For instance, all night stores have an individual location but are of the same address type. By using address types, users can easily configure settings for a large number of addresses. These can also be referred to as address kinds.

Case: A set of orders that have to be planned in their respective time windows on a set of available resources. A case is the input in its entirety as seen by ORD. ORD tries to find the planning for each case that has the lowest costs while fulfilling all constraints.

2.1.2 Configurable input data

In order to make ORD versatile and work with a lot of different real-life routing situations, it is highly configurable. A lot of flexibility comes from many types of restrictions that can be set, but since these will not be adjusted in any way, we will not discuss them. Also, the input data can be configured in multiple ways. The input data can be categorized in two main groups, namely: stop durations and travel times. Some other input data settings like (un)coupling durations and cleaning durations can also be configured but these are not the focus of this thesis and therefore will not be discussed. The reason for this will be explained in Section 2.2.

Stop duration

The stop duration is the time a vehicle is present at a location and is (un)loading orders. This can be either at a depot or at a customer. The time that ORD calculates the stop will takes is given by the following linear relation:

$$ST = FT + VT \cdot Q$$

Where ST, FT and VT indicate total stop time, fixed handling time and variable handling time respectively. Q indicates the quantity of products that is delivered in the stop, this is often measured in number of pallets. This means that to calculate the stop time, two input data values have to be set: the fixed handling-time and the variable handling-time. A pair of fixed and variable handling-times is called a handling-time setting.

Handling-time settings can be configured on several levels of specificity for groups or individual locations and products. ORD calculates the duration of a specific stop by searching for the most specific setting that is configured. The most common configuration is done on the level of the address-type. Then two handling-time settings are created for all stops occurring on a certain address-type, one for picking up orders and one for delivering. Another common occurrence is a setting for a specific product on an address-type. For instance, the user knows that when alcoholic products are delivered extra paperwork has to be signed, so he adds a handling-time setting for this product with a higher fixed time. Since separate handling-times can be set for

specific products and multiple products can be delivered in a single stop, ORD sometimes has to use different handling-time settings together to determine the duration of one stop. When this occurs, of all handling-times the highest fixed time is used and every variable time is used for each product quantity associated with that handling-time. While setting the handling time on the address-type level is common, it is also possible to configure handling time setting for individual locations. For practical reasons this is not often done.

Travel times

Travel times are an important aspect of routing problems. If these are not correctly set, then the solution provided by an optimisation algorithm might be far from optimal. In ORD, the travel times are calculated using maps. These maps contain information on the roads. To calculate the travel time between two locations, an optimization algorithm determines which roads are taken for the fastest route. The input for the calculations is the speed that the vehicles travel on certain road types. A set of speed configurations is called a vehicle profile. As with the stop times, the vehicle profiles can be configured on several levels. The most common implementation is to create vehicle profiles for vehicle types. Next to the configuration of vehicle profiles also a single number can be configured to increase or decrease all travel times.

2.2 Retailer case

In this research a learning method is developed and applied in a practical case at one of the customers of ORTEC. This section will describe the routing case of the Retailer and describe the available data.

2.2.1 Retailer

The Retailer is a large Russian retail company with thousands of stores and vehicles. The total region that is covered by the Retailer can be seen in Figure 2.1. Due to large distances some trips have total distances from over 1000 kilometres. These stores are supplied from 8 Distribution Centers (DC). ORD is configured such that these DCs have a complete separate implementation. This means that all DCs have a separate vehicle fleet, schedule, delivery area and settings.

2.2.2 Case

In the case study that will be performed the data of 1 DC is used. This DC delivers products to 873 stores. These stores are divided into five address types. In Table 2.1 the number of addresses and total number of stops per store-type can be found. On a single day, goods are delivered to on average 450 stores planned in 190 separate trips. To create feasible routes for this DC, ORD has to account for some constraints. These constraints are:

Vehicle capacity: The number of pallets that a truck moves cannot be larger the capacity of the vehicle. Different trucks can have different capacities.

Driving time legislation: After certain consecutive driving times, drivers are mandated to take a break before continuing driving.

Figure 2.1: Operating area of the Retailer

Table 2.1: The store types configured at the Retailer. The handling time per pallet for all of the address types is equal to 3 minutes.

Name	Number of individual addresses	Number of stops in dataset	Fixed handling time setting in minutes
SHOPRNDYD	712	124933	30
shop	107	16042	25
shop RND	49	12945	30
RND_45	4	1323	45
RND_60	1	395	60

Time windows: Orders have to be scheduled within a 6 hour time window that is predetermined. This way the store personnel knows when a delivery approximately arrives. During execution of the routes, the deliveries may deviate from these times as long as they are within the address opening times. When a truck arrives before a location opening time, then it will have to wait until it can start unloading. The time windows are separate from the address opening times, but will always occur when the address is open.
Depot loading capacity: The depots have limited loading capacity, this means that the departure of trucks has to be smoothed out over the day.
Product contamination: Some product types may not be shipped together in the same truck. For instance, fish and vegetables.
Vehicle capabilities: Some product types are chilled or frozen. These can only be transported by trucks with cooling capabilities.

2.2.3 KPIs

In daily operation the Retailer uses ORD to create routes and resources assignments to these routes to create feasible plannings. To provide a good set of routes, ORD makes use of hierarchical objectives to optimize the planning. This means, that a solution is accepted when a KPI improves when all KPIs that are considered to be more important remain equal. The other way around, less important KPIs may deteriorate as long as the higher ranked KPIs improve. The KPIs that

ORD uses, ranked on importance, are given below.

1. Number of planned orders
2. Plan costs
3. Distance
4. Total duration
5. Driving time
6. Number of trips
7. Waiting time

These KPIs all improve when they decrease, except for the number of planned orders. Since ORD uses heuristics and there are many restrictions and limited resources, it is not guaranteed that all orders can be planned. Therefore, the first goal of ORD is to plan as much orders as possible. The second KPI, the plan costs, is a weighted sum of other KPIs. The KPIs that are used in this sum are total duration, the distance, the number of stops and the number of trips. The weights of these costs can differ between different vehicle types. So, for instance, a larger truck has a higher costs per distance as it will use more fuel. These are configured so that they approximate the actual costs that are made when all orders are delivered according to the planning. The 3rd to 7th objective provide ORD with some tie breakers to decide on which solution is better when the plan costs are the same between two potential solutions.

Conversations with the Retailer have revealed that the total costs in their logistic operation consist of two parts. Besides the plan costs that are used by ORD, also some extra costs are made when the execution of the routes deviates from the planning. These costs, the service costs, consist out of costs that have to be made when time windows are missed. No clear definition exists for the service costs. The Retailer stated that only missing the time window is not overly problematic as long as the arrival is within the opening time of the shop. When the arrival at a shop is later then planned and outside the opening times of the store, the extra costs for that delivery are high.

2.2.4 Data description

The database that is used contains all planned routes for 1 year, from March 2018 to March 2019. This includes all information that is used by ORD to keep track of the entire planning and all constraints.

The data regarding the realised times of the actions is present only for the stop actions. For the stops the arrival and departure time at the locations is recorded. It is unknown whether the time of arrival is also the start of the unloading. Information on the travel actions can only be gained from the start- and end times of previous and following stops. This means that no information is present on the speed on certain road types or sections. For data regulation reasons all information regarding drivers, including anonymous identifier numbers, is not present in the dataset.

Since ORD has more detailed information regarding the stop and travels than only the total duration, a SQL-query was written to extract a dataset from the database that groups all relevant information on the stops and travels. This dataset has a single stop or travel action in every row and a feature in each column. All features and descriptions can be found in Appendix A.

2.2.5 Problem description

Using the knowledge we have on the dataset and current configuration of ORD at the Retailer, we will decide on which input data values to investigate and tune. Since both travels and stop duration can be controlled with parameters, these will be briefly investigated in order to make a choice.

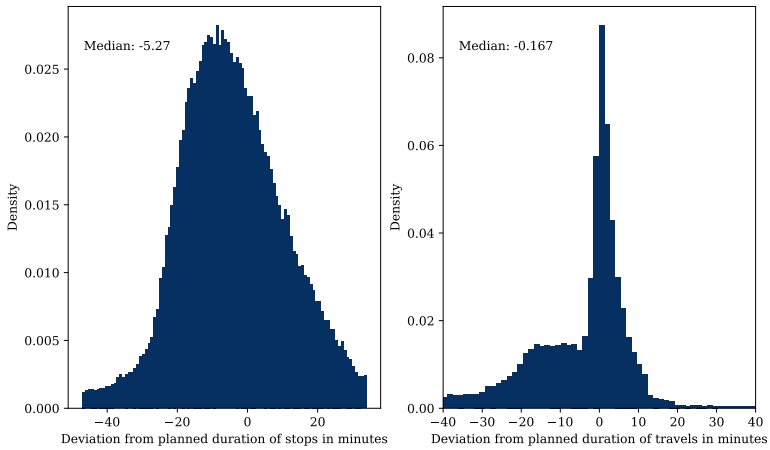

Figure 2.2: Histograms of the deviation from planned duration for stops and travels.

In Figure 2.2 the histograms of the deviation from the planned duration for both the stops and the travels can be seen. In this figure we see that the stops take on average shorter than planned. The travels deviate around the planned duration.

Since the travels seem to be configured well on average, and the data regarding the travel actions is not as detailed as the parameters that can be configured, we decide to only further investigate and tune the stop durations.

In Figure 2.3 the durations of the stop have been plotted against the number of pallets in the stop for each of the address types and the depot. This enables us to effectively plot the current configuration against what the relation is according to a linear regression. It is clearly visible, that in most cases the simple linear regression line already deviates from the current configuration. For the depot stops, we see that the data does not correspond at all with the setting that is configured in ORD. This means that there are probably actions occurring at the depot that are not configured in ORD. This is a problem that cannot be fixed only by data analysis. We therefore decide to not attempt to tune the stop duration set at the depot and not to use the data on the durations of the stops at the depot.

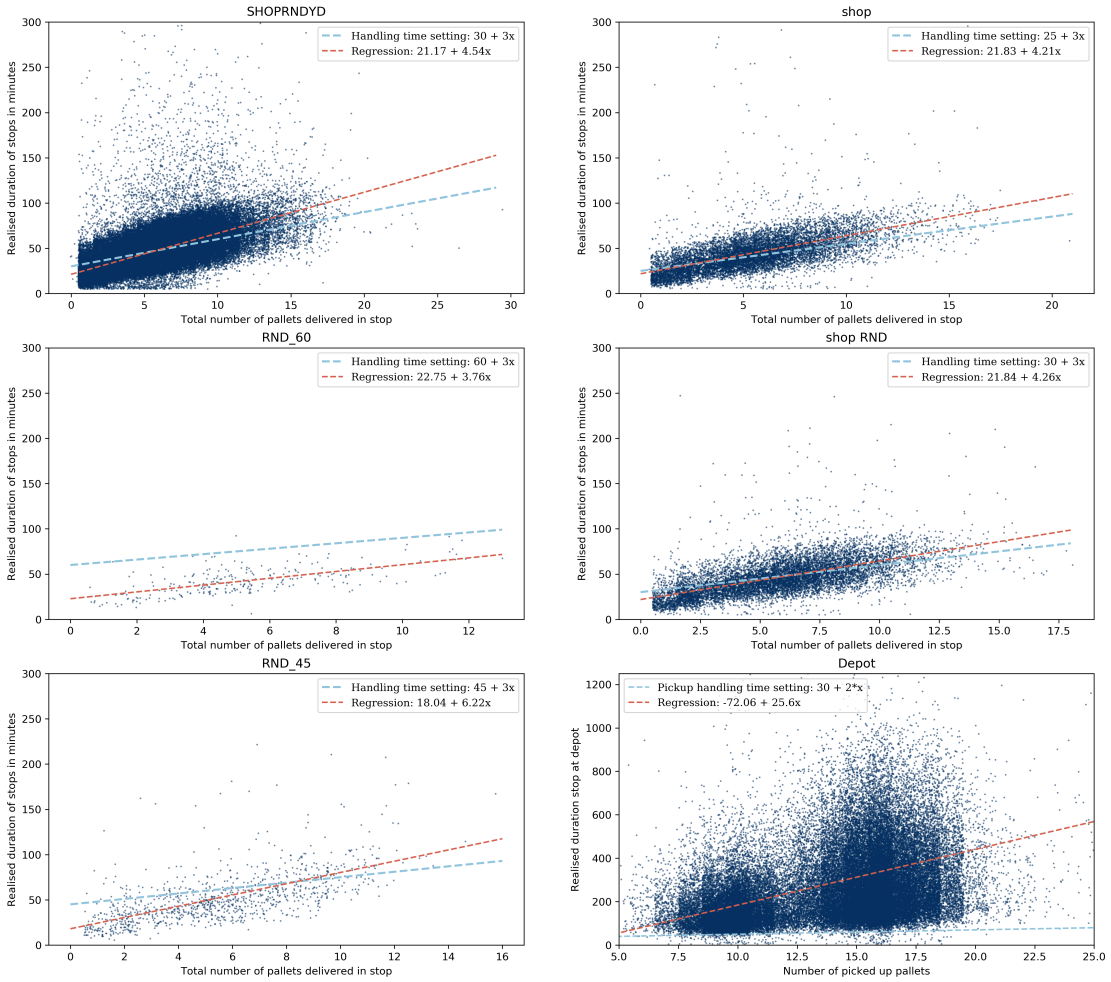

Figure 2.3: Scatter plots showing the relation between the realised duration of a stop and the number of pallets (un)loaded for every address type and the depot. The plotted lines indicate the current configured handling time setting and the relation obtained using linear regression.

The problem at the Retailer that will be solved by our learning method, is the tuning of the stop time settings of the five address types. Since the handling times at each of the address types consist of two durations, the total problem consist of finding ten continuous variables. Our learning method will try to find the ten values that give the lowest total cost. This total cost exists of plan and service costs. Both of these costs are affected by the changing of the handling time settings. The plan costs will be lower when the handling times are set to be shorter. When the stops take less time, the costs decrease since the total duration will be short. It may also occur that compared to using longer handling times, less trips or vehicles are needed to deliver all orders. The service costs however, will rise when the handling time is set to be shorter as more time windows will be missed. The costs of missing a delivery are not known exactly by the Retailer, meaning that these also have to be estimated somehow.

2.3 Conclusion

In this chapter, we have described the context in which this research is conducted. First, we have described what settings can be adjusted in ORD. Then, the characteristics of the routing case at the Retailer are described. The KPIs that are used to measure the performance of planning and realisation are given. Also, the data that is available has been described. From a brief investigation into the data and the current configuration at the Retailer, we have concluded that only the handling time will be adjusted and not the travel times. The handling time settings for five address kinds will be adjusted so that the total costs are minimized.

Chapter 3

Literature review

This chapter gives an overview of the available literature regarding the several fields this thesis touches. In the applicable subjects, methods will be chosen to implement based on the found literature. Section 3.1 first discusses the Vehicle Routing Problem (VRP). Then the VRP-variant most applicable to our case will be discussed. The available literature will be investigated on the usual methods for solving these. Next, the methods for determining or adjusting input data will be discussed. In Section 3.2 several important steps in the process of mining information from data are described. Section 3.3 gives an overview of several methods that can be used to estimate densities from data. Section 3.4 closes this chapter by describing some methods that are relevant for creating and optimizing a simulation model.

3.1 Vehicle routing problems

This section will first discuss the VRP and some variant of it. Then the limited research on the tuning of input data is discussed.

The VRP was first introduced by Dantzig and Ramser (1959) under the name: *The Truck Dispatching Problem.* It is a generalized form of the Travelling Salesman Problem (TSP) in which a set of customers has to be serviced by a set of vehicles given some constraints, while minimizing the total cost of the routes. The VRP is a NP-hard problem (Cordeau et al., 2002). Solving this problem to optimality using exact algorithms is possible to approximately 200 customers (Pecin et al., 2017). Therefore, in practice, mainly heuristics are used to solve these problems as realistic instances often are larger. The classical VRP is not representative for many practical situations since these often have multiple constraints.

To make the VRP more practical many variants of the original have been investigated. For instance, the Vehicle Routing Problem with Time Windows (VRPTW), is a variant on the VRP that constrains the delivery times to be within their respective time windows. There has been much research focussed on the VRPTW, as it is a constraint that needed for many applications. To solve this variant, (meta-)heuristics have been developed. The use of population-based search methods, large neighbourhoods and allowing infeasible solutions during the search seems to be successful in state-of-the-art heuristics (Toth and Vigo, 2014). The variant, in which a fleet of vehicles having different capacities is available, called the Heterogeneous Vehicle Routing Problem

(HVRP), was first studied by Golden et al. (1984). The HVRP does not only add a constraint, but may also alter the objective, as different vehicle types can have different associated fixed and routing costs (Baldacci et al., 2008). The goal is then no longer to simply find the set of feasible routes with the lowest total distance, but the routes that yield the lowest total costs are sought. A relaxation on the VRP, stating that deliveries may be split over several vehicles, named the Split Delivery Vehicle Routing Problem (SDVRP), was introduced by Dror and Trudeau (1989) to show that this could result in lower total costs. A survey on methods that have been developed and used to solve this variant, was given by Archetti and Speranza (2012).

Some other variants, are less well known as the constraints are not common in practice. An example of such a constraint, is the loading capacity of depots. As fleet sizes grow, one can imagine that there have to be imposed some constraints on the maximum number of vehicles that can be loaded at the same time. Rieck and Zimmermann (2010) modelled this constraint, and solved the problem using a Mixed Integer Linear Program (MILP) for a maximum of 30 customers. Another constraint of a practical nature is necessary when several product types, that have to be shipped in differing conditions, are loaded into several compartments within a truck. For instance, frozen, cooled and ambient temperature compartments. Ostermeier et al. (2018) investigated this variant, named Two-dimensional Loading and Multi-Compartment Vehicle Routing Problem (2L-MCVRP), and used a large neighbourhood search heuristic approach to solve problem instances up to 360 orders spread over 100 customers.

These variants mostly focus on adding one restriction at a time. However, real world applications often require several of these constraints. Giving every combination of a multi-constrained VRP a new acronym would become unclear, so these more realistic variants are given a more general overarching term, the "Rich"-VRP. Interested readers are referred to Toth and Vigo (2014) for more information on the numerous variants of the VRP and methods that have been developed to solve them.

In general, when more constraints are added or the number of customers increases, the problem becomes more difficult to find good solutions for. Recall, the problem at the Retailer was subject to all of the constraints mentioned in this section. Also, the problem size of 450 customers on average is quite large. This all results in a complicated problem that has to be solved by ORD. The heuristics that are used by ORD to solve this problem will almost certainly not find the optimal solution, and there are no guarantees on the distance between the optimal and the found solution.

3.1.1 Stochasticity

The routing problem of the Retailer may be modelled as deterministic VRP, but in reality we know that the problem has some stochastic components. Since we would like to improve solutions by using the information on these stochastic variables, we will briefly discuss the variant of the VRP that incorporates stochastic travel and service times, the Stochastic Vehicle Routing Problem (SVRP). Under the SVRP, variants exist where either the customers or demands are stochastic. We will however, focus only on variants where the times are stochastic.

Laporte et al. (1992) first introduced the SVRP, only considering the travel times to be stochastic. Here an a priori framework was considered, meaning that routes are planned and

are not altered during the realization of the travel times. A variant of interest of this research is the Stochastic Vehicle Routing Problem with Time Windows (SVRPTW) since we know that in the Retail case the travel- and service-times are stochastic and the total costs are determined by general routing costs and extra costs made when time windows are missed. The SVRPTW states that every customer must be serviced within a given time window and some aspects are stochastic. This can be further expanded by allowing the vehicles to wait outside a location for the start of a time window or continue servicing after the closure of the time window against some penalty. Variants considering multiple time windows are also present in literature.

A slightly different variant of the VRP also considers travel and service times to be uncertain. This variant, the Robust-VRP, assumes that there is uncertainty in some aspects of the input data and models this by using an uncertainty set instead of a distribution (Ordóñez, 2010). The use of a set of numbers to model uncertainty instead of a full distribution allows for easier calculations. The goal of these problems is to maximize the worst case scenario Hu et al. (2018).

Calculating the quality of a proposed solution for the SVRP can be a challenge for these problems. Since the models include stochastic parameters the value cannot be calculated exactly, only an expected value can be calculated. Wang and Regan (2007) provides some models to evaluate the quality of a proposed solution. This can be done for small instances by evaluating all possible outcomes with their probability and value. As instances get larger computing all scenarios quickly becomes intractable. For larger instances Wang and Regan (2007) leaves out the scenarios that are unlikely to simplify the calculation. For complex or large instances the value of a solution can be evaluated using a stochastic simulation (Li et al., 2010).

3.1.2 Input data

We will now discuss the limited literature regarding the estimation or adjusting of input data values for a VRP. Literature on the VRP typically describes a variant, the model and solution approach. How the input data values are determined is often missing in these descriptions. Also analysis on the consequences of altering input data values is not often discussed.

While variants like the SVRP or the Robust-VRP account for uncertainty in times, methods used to translate data from a real world problem into the needed distributions or uncertainty sets is not the focus of these research.

The research by Zunic et al. (2018), is the only one found that describes a method that was used to adjust input data of a VRP. Zunic et al. (2018) use GPS-data gathered by trucks to improve the prediction of service times and the location of the customers. Since the service time was given by a fixed time and a variable time per article the new prediction method also had to deal with that. They found that the prediction of the time per article was quit good. The fixed time was predicted at each customer by taking the average of the last 10 realised fixed times. These realised fixed times were calculated by taking the total realised time and subtracting the estimated time per articles times the number of articles that were delivered. In this research they only used the average duration that they measured, no consideration was given to the fact that these duration might vary more in some locations. Also, configuring a fixed service time slightly longer than the mean to create a more reliable planning was not discussed.

3.1.3 Conclusion

In the first section we have seen that the problem present at the Retailer is quite complex compared to current cases described in literature. The heuristics used by ORD will almost certainly not find the optimal solution and the quality of the solution cannot be determined. After that, we looked into VRPs that account for uncertainty in their model and optimization method. Using stochastic information during optimization is computationally expensive and cannot be done for large problem sizes. Calculating the quality of a set of routes, when uncertainty is present, is done using stochastic simulation when case sizes grow or are more complex. We found that there currently is a gap in research, on methods that can be used to find the right input data for VRPs. This thesis aims to shed some light on that part of the problem.

3.2 Data mining

This section will discuss some methods available in literature for gaining knowledge from data. In Section 3.2.1 data preprocessing steps are discussed.

3.2.1 Data preprocessing

Unfortunately data is most often not handed on a silver platter. It may be incomplete, inconsistent or noisy. There are several techniques that are able to deal with these issues. Maimon and Rokach (2010) give a good review on the steps that need to be taken to create a clean data set. The most important steps are:

- Missing value handling
- Feature selection
- Outlier detection

Missing value handling

Since most estimation methods or learning methods cannot deal with missing values the handling of these missing values is almost always the first step in the data pre-processing stage (Maimon and Rokach, 2010; García et al., 2016). The method that is chosen to deal with missing values can have a large impact on the results of the regression or classification problem the data is used for and thus should be chosen with care (Gromski et al., 2014).

The easiest way of dealing with missing values is list-wise deletion. This method simply deletes all rows or columns missing any values. The downside to this simple method is that it may cause important information to be deleted from the dataset.

In situations where deleting data is undesirable, using methods to fill in the missing values can be useful. This is called missing value imputation (García et al., 2016). Common ways are mean or median imputation. These methods typically simply impute the mean or median of the variable over the entire dataset for the missing variables.

Other more advanced methods use a learning algorithm to "predict" the missing value. This has been demonstrated to work using a k-Nearest-Neighbor algorithm or a Random Forest (Stekhoven and Bühlmann, 2012).

Feature selection

Dimensionality (the number of attributes in a dataset) can be a serious obstacle for data mining algorithms (Maimon and Rokach, 2010). This problem is also known as the "curse of dimensionality", as additional features increase the size of the search space for meaningful information exponentially (Elder IV and Pregibon, 1996). The process of feature selection helps to overcome this obstacle by reducing the dimensionality of the dataset. The goal of feature selection is to find the subset of features that give the best classification or regression model. Choosing the number of features to select is a trade-off between computational cost and model accuracy. In some cases the accuracy of the model even improves when redundant or irrelevant features are removed (Hall, 1999).

Feature selection techniques can be divided into filter methods and wrapper methods depending on the interaction with the learning process of the model (Maimon and Rokach, 2010).

Filter methods are heuristic methods that use general characteristics of the data instead of a learning algorithm to evaluate the added value of features. Correlation between the feature and the decision variable is a common example of such a characteristic. The use of simple characteristics of the data causes these methods to be computationally cheap, making them effective on high dimensional data. However, filter methods do not account for the interaction between the model and features. Also, correlation between several features is not taken into account.

Wrapper methods are methods that decide which subset of features to chose by training a model on subsets of interest. This means that these methods do account for the interaction between the features and the model. A disadvantage of wrapper methods is that the computational cost is a lot higher compared to filter methods since the models need to be trained multiple times.

Wrapper and filter methods can also be combined to select a suitable feature subset. This combined method uses a filter method to select the feature subsets check. The final decision on which subset to use is made by training the model on all selected feature subsets, as done for the wrapper methods. This combined method reduces the computational costs of a wrapper method by not testing every feature subset.

Outlier detection

The detection of outlying observations in the dataset is an important step in many data mining applications. Failing to clean erroneous data may lead to a biased estimation and incorrect results (Liu et al., 2004). A common outlier detection method, called the Standard Deviation (SD) method, is based on the assumption that the data is generated by a normal distribution $N(\mu, \sigma^2)$. Points are declared outliers by the SD method when they are outside an outlier region defined by a *confidence coefficient* α (Gudgeon and Howell, 1994). The α-outlier region of the $N(\mu, \sigma^2)$ distribution is defined by

$$region(\alpha, \mu, \sigma^2) = \{x : \frac{|x - \mu|}{\sigma} > z_{1-\alpha/2}\},$$

where z_q is the q quantile of the $N(0, 1)$. This boils down to considering points outliers when they are more than a predetermined number of times the standard deviation of the dataset from the mean of the dataset. However, this method is not robust since the estimated mean and

standard deviation can easily be influenced by the outliers (Hampel, 1985). It also relies heavily on the normality assumption which is violated in skewed or heavy tailed distributions.

The Median Absolute Deviation (MAD) method is a robust version of the SD method (Leys et al., 2013). Where the SD method uses the amount of standard deviations from the mean to set the outlier region, the MAD method uses the amount of MAD's from the median of the data set. The MAD is a measure of statistical dispersion that can be calculated for univariate data set $x_1, x_2, ..., x_n$ as follows.

$$\text{MAD} = median(|x_i - median(X)|)$$

This measure is less easily influenced by the outliers than the standard deviation. The MAD method can be extended for skewed distributions by calculating the MAD for data points higher than the median separately from the MAD for data points lower than the median.

3.2.2 Conclusion

To create a clean dataset all three steps will be performed. To handle the missing values, we choose to use list-wise deletion as the size of the dataset allows this. This method is preferred over value imputation for implementation ease. The method used for feature selection will be a combined wrapper and filter method. This choice is made because a combined method reduces the computational cost needed for a pure wrapper method and increases the accuracy of a pure filter method. To identify and remove outliers from the data set, we choose to use the extended MAD method. This method is chosen for the improved robustness over the SD method. The extended version is chosen to deal with non symmetrical data sets.

3.3 Density estimation

In order to use data in a simulation, probability densities have to be inferred from this data. This common statistical problem is called Density Estimation. When a dataset is present with some explanatory variables and the goal is to estimate the density of the dependent variable when the explanatory variables are known, the problem is referred to as Conditional Density Estimation (CDE). Given a data set, CDE aims to describe the statistical relationship between a conditional vector $\mathbf{x}$ and a dependent variable $\mathbf{y}$ by modelling the conditional probability $\mathbf{p(y|x)}$.

3.3.1 Statistical methods

The most common methods for the fitting of distributions to data come from the field of statistics. The distributions that are fitted can be divided into theoretical and empirical distributions (Law, 2011). These are explained further below. The underlying assumption for any distribution is that all data that is observed is independent and identically distributed (IID). The data is divided in sets that are all assumed to be generated by a common distribution. This can be done by separating the data on a categorical variable for which it is either known or seen that it influences the dependent variable.

Theoretical distributions

Theoretical distributions are probability density functions that can be parametrized. Some examples are the Gaussian and Gamma distributions. When a theoretical distribution is fitted on a data set, random samples can be drawn from that distribution and used for simulation.

The first step for finding a distribution that represents the data well, is to decide what general families (e.g., Gaussian, lognormal, exponential, gamma or Weibull) seem appropriate based on their shape. A histogram of the data is a good tool for determining the shape of the data (Law, 2011). The second step in the process, is the estimation of the parameters for the selected distribution families. There are several methods for estimating these parameters. The two most common methods are the method of moments and the maximum likelihood estimation method (Montgomery and Runger, 2003).

To choose between the distributions that have been fitted in the second step, the fit of each of these distributions has to be checked to see how representative they are. This can be done graphically or by performing a goodness-of-fit test. A graphical approach is to create a density-histogram plot. A goodness-of-fit test is a statistical test to test the null hypothesis H_0 that the data is generated by the fitted distribution. A common example of a goodness-of-fit test is the χ^2-test (Law, 2011). The χ^2-test is common because it can be applied to any univariate distribution. Another goodness-of-fit test is the Kolmogorov-Smirnov test. This test measures the largest distance between the Cumulative Distribution Function (CDF) of the fitted distribution and the empirical distribution function of the data. The Kolmogorov-Smirnov test has generally a lower error rate than the χ^2-test and is therefore more valid in general (Hsiao-Mei, 2009).

An advantage of using theoretical distributions to represent the uncertainty in a simulation model is that they are computationally cheap to fit and use during simulation. They also take up little memory since the dataset is represented by only a few parameters. Theoretical distributions also work with small amounts of data and give smooth distributions that extrapolate the probability mass outside the observed data (Law, 2011).

A disadvantage is that most distributions are only defined for the univariate case and cannot be used in a multivariate setting. Also the data has to be split into groups that are assumed to be generated by the same distribution. No data is shared between these groups, which may cause some information to be lost. Also the groups have to be defined by the user based on some prior knowledge.

Empirical distributions

Empirical distributions are distributions that are fully defined by the data set that was used to create them. A simple method to create such a distribution is to create a continuous, piecewise-linear empirical distribution function (Law, 2011). This can be done by sorting the data set so that $X_{(1)} \leq X_{(2)} \leq ... \leq X_{(n)}$. We can then define $\tilde{F}(x)$ as follows :

$$\tilde{F}(x) = \begin{cases} 0 & \text{if } x < X_{(1)} \\ \frac{i-1}{n-1} - \frac{x - X_{(i)}}{(n-1)(X_{(i+1)} - X_{(i)})} & \text{if } X_{(i)} \leq x \leq X_{(i+1)} \text{ for } i = 1, 2, ..., n-1 \\ 1 & \text{if } X_{(n)} \leq x \end{cases} \tag{3.1}$$

An example of such an empirical distribution can be seen in Figure 3.1. This method gives a rough CDF when the number of observations is low, but will result in a smooth distribution when a large enough data sample is used. This method can be extended to accept weighted observations (Kaczynski et al., 2012).

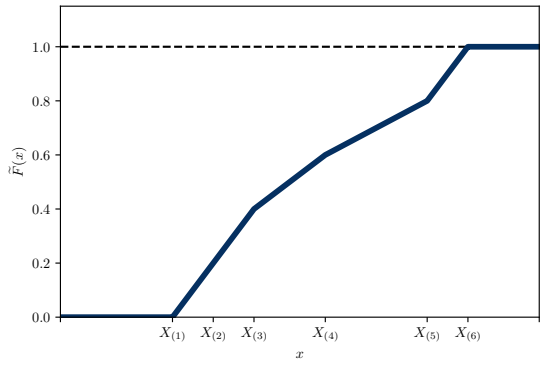

Figure 3.1: Example of a continuous, piecewise-linear empirical distribution function.

A more advanced non-parametric method to estimate densities is Kernel Density Estimation (KDE). This method has a strong relation with the histogram but is able to give a more smooth density estimate by using the kernel and selecting the right bandwidth (Silverman, 1986). Where a histogram divides the range in a finite number of bins and stacks the density in the bins if a point falls into them, the KDE assumes the probability around each point is given by a kernel function. The density estimate is then the sum of all individual kernels. An example of the relation between a histogram and a KDE can be seen in Figure 3.2. The kernel is a non-negative function parametrized with a smoothing parameter often referred to as the bandwidth. The selection of the bandwidth has a large impact on the resulting estimated density. This effect can be seen in Figure 3.3. If the bandwidth is chosen too wide the resulting density is over-smoothed, picking a too narrow bandwidth gives a under-smoothed density. The bandwidth can be selected using some rule of thumb or by using cross validation to find the bandwidth that gives the highest likelihood. Using rules of thumbs is faster than using cross validation but can give over-smoothed results when the underlying assumptions on normality are violated (Botev et al., 2010). KDE can be extended to estimate densities in a multivariate setting (Scott and Sain, 2004). This means also conditional density estimation can be done. However, this assumes a spatial relation between all variables which may not hold when some of the input variables are categorical.

An advantage of empirical distributions is that no assumptions have to be made on the shape of the underlying distribution and that the empirical distribution will converge to this underlying distribution when more observations are added.

A disadvantage of empirical distributions is that they have difficulty extrapolating the probability density outside the bounds of the observed data. This is especially a problem when the number of observations is low (Law, 2011). When the number of observations is high this is not a problem, but memory related issues can occur. Unlike theoretical distributions, empirical distributions are defined by all observations. So when an empirical distribution is used in a simulation model, all observations must be held in memory during the sampling of random variates.

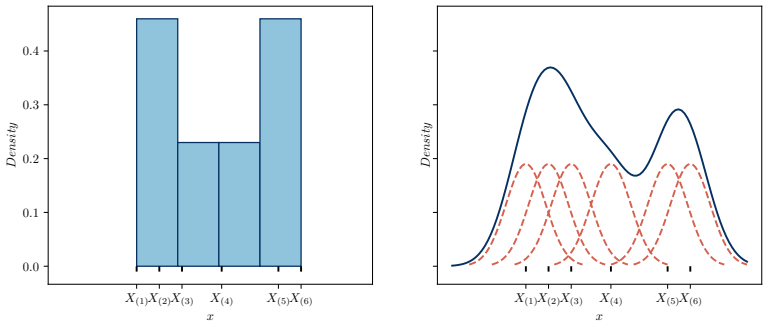

Figure 3.2: Left: histogram on 6 datapoints. Right: kernel density estimate on the same 6 datapoints. The red dashed lines show the individual kernels, the blue line the kernel density estimate.

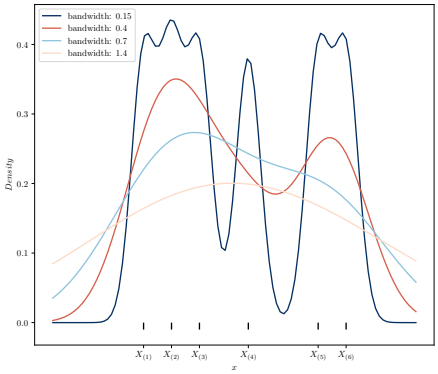

Figure 3.3: KDE with different bandwidths.

3.3.2 Advanced statistical methods

Methods to estimate probability densities have been developed in the machine learning community by transforming or adapting existing statistical methods. These methods are still statistical methods, albeit an advanced form. We put them in a separate category because, unlike the previously listed statistical methods, the models that these methods create cannot fitted in an exact and optimal fashion. These models consist of a large number of parameters that are found using heuristic methods during a training phase. These methods are especially useful when dealing with large datasets containing mixed data types. These methods come, like the statistical methods, in parametric and non-parametric flavours. The two most well known methods that have been combined to estimate densities are Neural Networks and Random Forests. Both Neural Networks and Random Forests can be combined with either flavour. While more methods exist, these two are investigated for their ability to deal with non-linear relations between features and distributions.

Neural Networks

A Neural Network (NN) is an, often layered, collection of connected perceptrons that map input signals to output signals by receiving, non linearly processing and propagating these signals.

Mixture Density Networks (MDNs) combine NNs with a mixture density model in order to estimate conditional probabilities (Bishop, 1994). A NN is trained to map the input vector $\mathbf{x}$ to the parameters θ_i and mixture coefficients α_i of a set of K mixture components. The basic shape of the components is set to be Gaussian. The network is trained to so that the log likelihood is maximized. The conditional density of a sample as computed by the network is given by:

$$p(y|\mathbf{x}) = \sum_{i=1}^{K} \alpha_i(\mathbf{x}) p(y|\theta_i(\mathbf{x}))$$

Ambrogioni et al. (2017) combines parametric with non-parametric methods to create Kernel Mixture Networks (KMNs). As with MDNs, KMNs utilize a NN to map the input vector $\mathbf{x}$ to the mixture weights α_i. The location and scale of the mixture components are not determined by the NN however, these are, like with KDE, set at the location of the training points. To control the size of the model, the number of used kernels can be limited by clustering the training points.

Since the location of the mixture components is fixed for the KMNs, the flexibility is lower than of the MDNs. It does however, reduce the risk of overfitting compared to MDNs (Rothfuss et al., 2019). But still these methods are both quite prone to over-fitting when training the models to maximize the likelihood. The use of proper regularization techniques is therefore important. These models are both shown to generalize better when they are trained with noise added to the training sample labels (Bishop, 1995; Rothfuss et al., 2019).

Random Forests

A Random Forest (RF) is an ensemble of decision tree regressors such that each tree is trained on a random bootstrap sample of the data (Breiman, 2001). A decision tree is a directed graph that can be used for regression or classification by dividing the total feature space into a finite number of rectangular clusters and predicting labels for each cluster (Breiman et al., 1984). To create a decision tree, the features and thresholds to split on have to be chosen. Since finding the optimal tree is computationally impossible due to the large number of combinatorial alternatives, heuristic methods are employed to recursively train trees. During training the split is chosen that creates the two nodes that most improve some loss function indicating node impurity (Zaman et al., 2011). Node impurity is a measure of homogeneity of the samples in the node, thus pure nodes indicate that all samples in the node are similar. Besides the randomness introduced into each tree in a random forest by taking a bootstrap sample of the training data, RFs also introduce variation by examining only a random subsample of features for the best split during training for each split (Ho, 1995). The introduced variation among trees within a RFs overcome the tendency of individual decision trees to over-fit (Kleinberg, 1996).
Conditional quantiles can be predicted by a generalization of RFs, called Quantile Regression Forests (QRF) (Meinshausen, 2006). This was achieved by recognizing that RFs actually provide an estimation for the conditional mean $E(Y|X = x)$ by calculating a weighted mean over the observations of the dependent variable Y in the training set. X here is the random vector containing all independent variables from which x is a sample. Following the notation of Meinshausen (2006) it is shown that a tree makes a prediction for a new sample x by taking a weighted average of the training sample data. The weight w_i for training sample i predicted by

tree t is defined as:

$$w_{i,t}(x) = \begin{cases} 0 & \text{if } i \text{ not in same leaf as } x \\ \frac{1}{\#(\text{samples in leaf})} & \text{if } i \text{ in same leaf as } x \end{cases} \tag{3.2}$$

The prediction of tree t for sample x when trained on training data $y_i; i = 1, 2, ..., n$, is then given by:

$$\tilde{\mu_t}(x) = \sum_{i=1}^{n} w_{i,t}(x) y_i \tag{3.3}$$

The prediction of a random forest is calculated by taking the average weights for the training samples predicted by the trees:

$$w_i(x) = \frac{1}{k} \sum_{t=1}^{k} w_{i,t}(x) \tag{3.4}$$

The prediction of a RF is then:

$$\tilde{\mu}(x) = \sum_{i=1}^{n} w_i(x) y_i \tag{3.5}$$

QRFs use these predicted weights of the training samples in combination with the training sample to fully specify an empirical distribution of the newly predicted sample. This is done by using an adaptation of the piecewise-linear cumulative distribution function, Equation (3.1) so that weighted samples are accepted.

Pospisil and Lee (2018) improve on QRF by using a more appropriate impurity function on which to test the splits during training. Their method, named Random Forests for Conditional Density Estimation (RFCDE), replace the Mean Squared Error (MSE) impurity used by QRF with a impurity function specific for CDE (Izbicki and Lee, 2017). Where MSE detects only changes in the mean of a node the new impurity is sensitive to changes in the empirical distribution.

Random Forest can also be used for the parametric estimation of densities. Cousins and Riondato (2019) introduce RFs that estimate the parameters of a theoretical distribution called 'CaDET'. The distribution family has to be chosen prior to the training of the model. At the leaves of the tree only the sufficient statistics have to be stored instead of all training samples. The use of only some summary statistics in the leaves instead of all training samples results in a model that is smaller in size and faster to query compared to QRF and RFCDE. However, being a parametric method CaDET will be outperformed by RFCDE or QRF when the shape of the data does not nicely follow a theoretical distribution.

3.3.3 Evaluating predictions or distributions

Since different methods can be applied to estimate the distribution of data points, a method to compare different distribution models is needed. When choosing a distribution model, it is desirable that the model explains the data well. Cox (1961) gives a test statistic to compare 2 non-nested models. This test statistic uses the likelihood of the data under the given models. The likelihood of a single data point under a model m with parameters θ is:

$$\mathcal{L}(\theta, m|x) = p(x|\theta, m)$$

When this test statistic is calculated for a data set the joint probability of observing the data set under the distribution is calculated by:

$$\mathcal{L}(\theta, m|\mathcal{D}) = p(\mathcal{D}|\theta, m)$$

where the probability of the dataset can be calculated by:

$$p(\mathcal{D}|\theta, m) \overset{\text{i.i.d.}}{=} \prod_i p(d_i|\theta, m)$$

For computational reasons it is common to work with the natural logarithm of this number since the product of many probabilities will quickly lead to a very small number, which may cause numerical underflow. This gives:

$$log(\mathcal{L}(\theta, m|\mathcal{D}) = log(\prod_{i=1}^{N} p(d_i|\theta, m))$$

$$log(\mathcal{L}(\theta, m|\mathcal{D}) = \sum_{i=1}^{N} log(p(d_i|\theta, m)))$$

This can be used for any type of model to determine the likelihood of a dataset under the model. Since this likelihood will always only decrease when the data set grows, it is appropriate to normalize this value for the size of the data set used to determine it. It is also easily extensible to a conditional version, the measure is then called Average Conditional Log Likelihood (ACLL) (Norwood et al., 2001). Assuming the m parametrized by θ is tasked with predicting the density of Y given X, this measure is:

$$ACLL = \frac{1}{N} \sum_{(x,y)\in\mathcal{D}} log(p(y|\theta, m, x)))$$

Simply using this measure and training or fitting models to find the maximum value may lead to a model that perfectly explains the data on which it was trained, but fails to generalize (Hawkins, 2004). This is called over-fitting. Since the aim is to create a model that finds the underlying distribution it is desirable to have a model that performs well on data samples that it has not yet seen. Different methods have different ways to account the performance measure for over-fitting. When statistical models are used, typically a penalizing measure that is related to the number of parameters is used. For RFs, performance can be measured for each tree on the samples that are not used in the training of that specific tree, the so called out-of-bag samples. A common method that is used to check the generalization of machine learning models is to separate the data set into a training and test set and train the model exclusively on the data present in the training set. The final decision of model choice is based on the accuracy (e.g. likelihood or MSE) of the test set containing data that the model has not seen before (Maimon and Rokach, 2010). This can be extended to an estimator that uses all the data by using cross validation (Smyth, 2000). Using cross-validation has the advantage of being generally applicable to any model or accuracy function. A disadvantage however, is that it may take a large amount of computational power in order to train a model on every fold.

3.3.4 Conclusion

There are many methods available to estimate conditional densities. In this thesis these methods will be used to provide density estimates to be used in the simulation model. We decide on which density model to use based on the cross-validated ACLL. We will use statistical methods and advanced statistical methods and compare these to see which perform better. The advantage of the statistical methods lie in their speed and interpretability. Given the size of the data set, the advanced statistical methods are expected to perform better because of their ability to use many explanatory features in a non-linear fashion. Among the advanced statistical methods described we firstly chose in favour of a RF based method because these are generally easier to implement, train and tune in comparison to NN methods. Among the RF methods we opt for the RFCDE method developed by Pospisil and Lee (2018) because of the fully non-parametric nature of it.

3.4 Simulation

This section describes methods known in literature for setting up and verifying simulation models in Section 3.4.1. In Section 3.4.2 methods regarding the optimization of simulation models are described.

3.4.1 Building and verifying a simulation model

Setting up a simulation model is the process of gathering information on the system structure and operating procedures and translating this into a computer program (Law and Kelton, 2015). This model can then be used to test a set of interventions in order to make a decision on the implementation of one of the interventions. In order to draw valid conclusions it is important that the simulation model is a reasonable model for the real process. This can be established by carefully building and validating the model.

Building a simulation model is done by first creating a conceptual model and ask people involved in the real life process to check this. Law and Kelton (2015) recommend to keep a documented list of assumptions made for the model. This can be divided into assumptions on the components of the system and the interactions between these components (Banks et al., 2010). Also the input parameters and data assumptions have to be checked, this is done by validating the distributions that are used as sampling source. This conceptual model can then be converted into an operational simulation model.

The validation of a simulation model can be done in several ways. Law and Kelton (2015) suggest several techniques that are applicable to our case. The first and foremost technique used to verify a model is to check the code of the model. For larger models it is useful to check this per code module and use more than one person to check these parts (Law and Kelton, 2015). Other important techniques involve the checking of the inputs and corresponding outputs of the model. First the model can be checked to see if the outputs for various inputs are reasonable. Next the model can be run with a set of input values for which an approximate output value is known to see if the model adequately approximates the reality (Sargent, 2011).

3.4.2 Simulation optimization

Simulation optimization is attempting to optimize a problem where the relation between the input and output is unknown and there is uncertainty present (Amaran et al., 2016). This uncertainty comes from the fact that the output is generated by a simulation. This can be mathematically formulated as:

$$\arg\max_{x \in \mathcal{X}} E[f(x)]$$

There are many available optimization techniques for many different problems. We know already a couple of aspects of our problem that help limit the search for applicable optimization methods. For instance we know that our function $f(x)$ is a simulation model and thus no derivatives are available. We also know that the function is stochastic, resulting in noisy measurements. Luckily, we know that all attributes of the input vector x are numerical instead of categorical making it possible to use methods utilizing some distance metric between input values. Since our function evaluation is a computationally expensive simulation run, we focus our search on methods that sequentially decide which input vector to evaluate next in order to find a near-optimal solution in few evaluations. Sequential methods applicable to simulation optimization can be roughly divided into two categories: gradient approximation methods and Response Surface Methodologies (RSM).

Gradient approximation

The Finite Difference algorithm approximates the gradient with respect to the input by computing an approximation of the gradient using a pair of simulations for every input variable (Spall, 2005). To optimize the simulation the next input set is then chosen by performing a gradient descent step. This method requires a growing number of function evaluations as the dimension of the input grows. Simultaneous Perturbation Stochastic Approximation tackles this problem by using only two simulation evaluations to estimate gradients of all input variables (Hill and Fu, 1995; Bhatnagar et al., 2013). These methods are proven to converge to a local optimum and are easy to implement. However, they may take many iterations to converge (Kim, 2006).

Response Surface Methodologies

RSM focus on learning the relation between inputs and outputs by approximating this relation with a functional surface, also known as surrogate model(Law and Kelton, 2015). This surrogate model can then be used to find the inputs to be simulated next. The value of a new potential sample point is determined by an acquisition function that uses the surrogate model, such that the next point is sampled at the argmax of the acquisition function (Brochu et al., 2010). Methods in RSM differ on the surrogate model, acquisition function and how the surrogate model is updated (Amaran et al., 2016). Most methods are developed for stochastic functions, but some are used specifically for deterministic measurements. The Efficient Global Optimization (EGO) method fits a smooth function through deterministic measurements and uses the expected improvement to select the next points (Jones et al., 1998). The expected improvement is also used in Bayesian Optimization (BO), a generalization of EGO, which uses a Gaussian process as surrogate model (Frazier, 2018). A Gaussian process is a generalization of a probability distribution (which describes a finite-dimensional random variable) over functions such that every

linear combination of these random variables follows a Gaussian distribution (Rasmussen and Williams, 2005). These Gaussian processes can be efficiently sampled using a method called kriging to obtain an interpolation between known values (Jones et al., 1998). This can then be used to efficiently calculate the values of the acquisition function. This can be used in combination with techniques like the Upper Confidence Bound (UCB) policy, which searches the point with the highest UCB to sample next(Chang et al., 2007). Other policies include Expected Improvement (EI) and Probability of Improvement (PI), these polices respectively maximize the expected value of the improvement in the next sample point or the chance that an improvement is observed. Where the PI-policy can get stuck in local optima, the EI-policy offers a better trade-off between exploitation and exploration by also considering the size of the improvement. These three policies are not necessarily restricted to the use of a GP surrogate model.

A subset of BO that aims to reduce the number of samples that have to be taken in order to optimize a problem is Optimal Learning (Powell, 2010). Methods in Optimal Learning focus mainly on the acquisition function. In Optimal Learning, online and offline learning are separated by the fact that an offline learning problem has a separated learning and implementation phase, while during online learning the implementation and learning happen simultaneously. This means that for online learning the total reward over all measurements has to be maximized, while in an offline learning case the best input has to be found in a limited number of measurements and the reward during the learning process is irrelevant. Some simple heuristics are applicable in on- and off-line learning (Frazier et al., 2008). A more advanced method, the Knowledge-Gradient (KG) policy (Frazier et al., 2008), was originally intended for offline learning but can be used for the online case with a small adjustment (Ryzhov et al., 2012). This policy is not necessarily restricted to using a Gaussian-process as surrogate model. The concept behind the KG is to sample the points that reveal the most information on all points. To calculate the expected increment in the value of information Frazier et al. (2008) assume a priori knowledge on the correlation in the surrogate model. This can be difficult to obtain when using other surrogate models than a Gaussian process. The Hierarchical Knowledge-gradient only needs a known a priori structure of the input space, which is easer to obtain in general (Mes et al., 2011). The most used policy is the Expected Improvement policy since it performs well and is easy to implement (Frazier, 2018). The Knowledge-Gradient policy is more complicated to implement but performs better in more complex settings (van der Herten et al., 2016).

Finding the next sample point in BO is done by finding the point that maximizes the acquisition function. This however, is in itself a optimization problem. This is normally solved in one of two ways: sampling random points and taking the maximum or using a gradient ascent algorithm. The sampling of random point is easy to implement since only the acquisition function is needed. It can become inefficient however when the input is high-dimensional. Using a gradient ascent algorithm can be more efficient in high dimensional space, but is more involved to implement since besides the acquisition function also the derivative of the acquisition function is needed. Since gradient ascent methods only provide local optima, a multi-start version is typically used to find the global optimum (Frazier, 2018).

3.4.3 Conclusion

The methods known in literature on building and verifying a simulation model all give the same general way to do this. In this study we will also apply these steps when building the model. For the optimization of the simulation model many different methods exist. Since we already know that performing a simulation run will be computationally expensive, we want to use a method that uses a small number of simulation runs to come to a near optimal solution. For this reason the gradient approximating methods are not suitable. Methods developed in the BO field seem like a good fit for the optimization problem present since these are able to perform global optimization on continuous and high-dimensional inputs where the outputs are noisy. Only the choice of acquisition function remains. We chose to implement the Expected-Improvement policy for the decent performance and easier implementation compared to the Knowledge-Gradient policy.

Chapter 4

Case Study

This chapter describes the steps that were taken in the practical application of our method on the case of the Retailer. Section 4.1 explains the methodology. In Section 4.2 the steps taken to create a clean data set are described. In Section 4.3 the process of estimating distributions and choosing the best model is given. Section 4.4 explains the setup of the simulation model and the assumptions that have to be made.

4.1 Methodology

To elucidate the steps that are applied in this thesis, we will explain these steps and their order in this section. In Figure 4.1, all steps that will be performed can be seen. These have been divided into four phases. The first three phases of the methodology are handled in the remainder of this chapter. The last phase of the methodology is applied in the next chapter.

In the first phase, the data preprocessing phase, the data from the database will be transformed into a cleaned data set. The input for this phase is a snapshot of the database of the Retailer. To do this, first the raw data needs to be extracted from this database. Data from multiple tables within the database have to be joined in order to create a single data set. This process is mixed with the next step, feature creation. Feature creation will add calculated and derived features that are not originally in the dataset but are expected to be useful based on expert knowledge. When these two steps have been completed an iterative process of data checking and cleaning will be performed. The cleaning of the data will consist of handling the missing values and removing outliers according to the method found in Chapter 3. The output of this phase is a clean data set.

In the second phase, the cleaned data set will be used to estimate the distributions needed for the simulation. First, we determine what sources of randomness exist in the real world. A distribution or distribution model is needed for every source of randomness. These distributions will be estimated using a statistical fitting method and an advanced statistical method. For both these methods, the explanatory variables influencing the dependent variable will be investigated first. For the statistical method, distributions will then be fitted to the data set, and the best fitting distributions family is chosen among these fits. For the advanced statistical method, feature selection will first take place to reduce the number of variables.

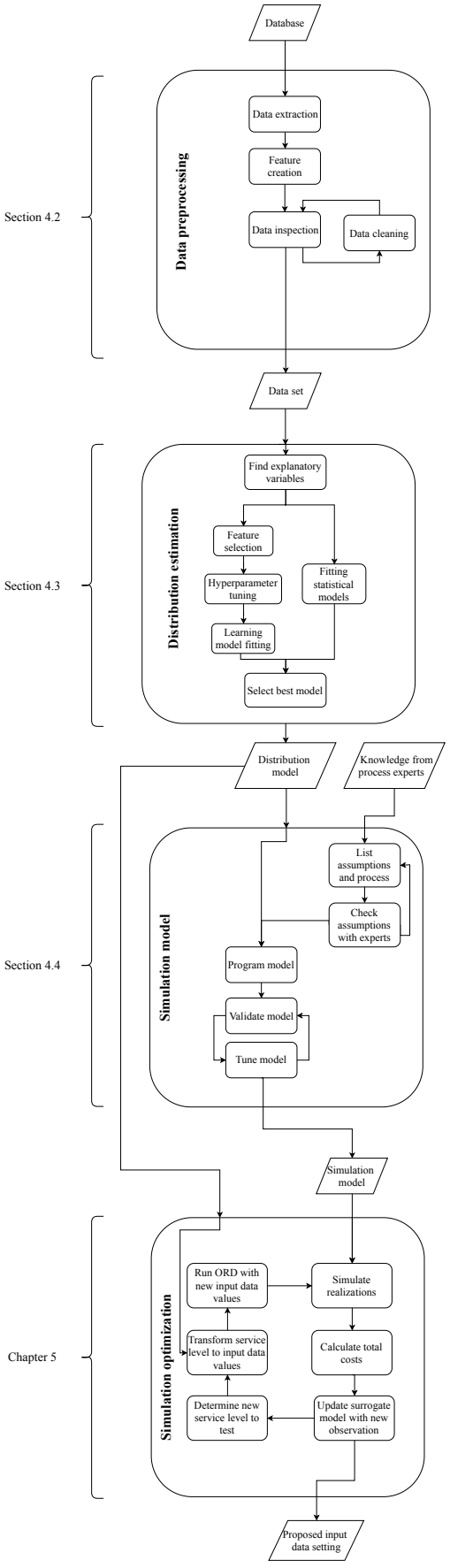

Figure 4.1: Flowchart of the methodology used in the remainder of this thesis.

After a subset of features is chosen, the hyper-parameters of the model will be tuned to get a good performing model that does not overfit. To choose the best model among the statistical and advanced statistical models, the models will be compared on performance using cross-validation.

When the distributions have been estimated from the data, the simulation model can be built. Besides the distributions, also some knowledge from process experts is needed in this step. This knowledge will be used to formulate a list of assumptions that lie at the heart of the simulation model. These assumptions combined with the distributions will be programmed into a model that will simulate trip executions. To check the validity of this model, we will compare simulated realised durations with historic realised durations. This will be done by using the historic planning to create a set of routes as input for the simulation. Then the historic realised durations from the data set can be compared to the simulated realised durations when the planning is the same. This validation process can be used to tune the model in order to obtain the desire level of accuracy.

In the final phase, the simulation model will be used to find the optimal settings. To simplify the optimization, we reparametrize the problem. As mentioned in Section 2.2.5, the goal is to find a handling time setting for all five address kinds. Instead of searching for the five sets of fixed and variable handling times, we search for a single value that represents the fraction of stops that is completed within their planned duration. We use our distribution model to find the input data values that should be configured for any given input data percentile. As determined in Chapter 3, a Bayesian Optimization (BO) algorithm will sequentially decide which percentile of the distribution of the input data values to test. In the test of a given percentile several steps are performed. First, the percentile will be used to calculate all ten input data values to be used in the planning optimization. Then, using these input data values, a planning will be made for a predetermined selection of cases. These plannings will be used in a simulation to determine the service costs made by missing time windows. The service costs combined with the plan costs calculated by ORD make up the total costs. These total costs are used by the simulation optimization algorithm to determine the next percentile to test.

The output of the last phase is a proposed new set of input data values to be used. This will be compared to the current settings on more tangible KPIs to assess the value of this new setting.

4.2 Data preprocessing

This section describes the different steps that were taken to create a clean dataset. First, the handling of the missing values is described. Then, the methods used for removing the outliers are explained.

4.2.1 Missing value handling

To create a dataset that can be used for distribution estimation, first the data records that are missing values should be dealt with. As described in Section 3.2.1, the options are to either remove data points missing values or impute the missing values.

Upon inspection of the data set it was seen that mainly columns regarding the realisation

data contained missing values. In Table 4.1 the number of missing data values can be seen. Since the remaining data set is still of reasonable size we decide to use list wise deletion on the data entries that are missing values and simply remove these from the data set.

Table 4.1: Variables in data set missing values

Variable	**Number of missing values**	**Percentage missing values**
realisedStartInstant	17852	3.82%
realisedFinishInstant	18798	4.03%
realisedDurationMinutes	23726	5.08%
id_unionResourceCombi	385	0.08%
Any of the above	23789	5.09%

4.2.2 Data Cleaning

The cleaning of the data set is done in two steps:

- Visual inspection for aberrant data
- Statistical outlier removal

The data is first inspected using some summary statistics and visualizations. This way we can quickly detect values which we know are incorrect and remove these. Then a statistical procedure is used to determine which data points are outliers and these are also removed.

Visual data inspection

During inspection using summary statistics, it was found that some variables, which we know are non-negative, contain negative values. For instance it is known that all variables that contain any form of durations have to be positive. The entries containing negative values are removed from the dataset.

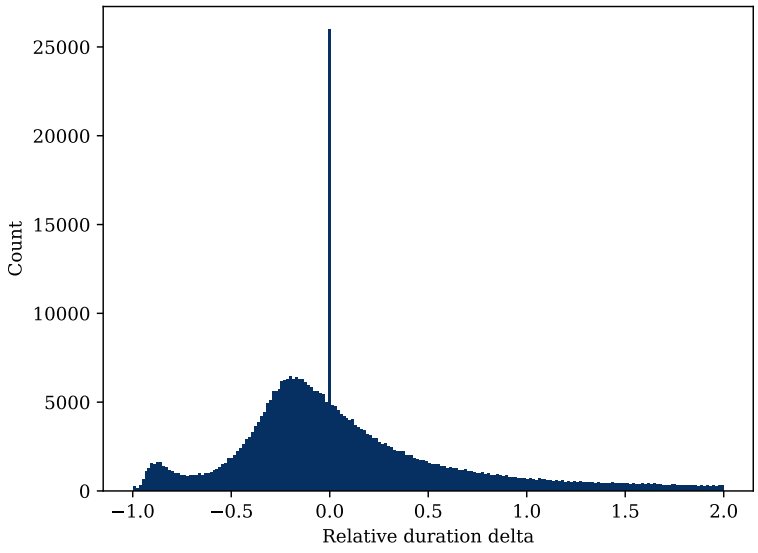

Figure 4.2: Histogram of the relative duration delta

After inspecting the summary statistics some visualizations are investigated for strange values. In Figure 4.2 one of these visualizations can be seen. It is clearly visible that there is a peak of data points having a relative duration delta of 0. Further inspection revealed that these data entries had a delta of exactly 0 and the planned start and finish times exactly equal to the realised start and finish times. This is a clear indication that either the planned or the realised times of these data points are incorrect. All data points that have a delta of exactly 0 have also been removed from the data set.

Statistical outlier removal

The removal of outliers is essential when working with real world data. We will be removing outliers by looking at one variable at a time. We opt for this univariate removal of outliers because it is fast and easy to perform. Before we can remove outliers, we have to split the data set into subsets for which we can assume that the variables of interest are produced by the same underlying distribution. This firstly means that we will look at the stop and travel actions separately. The features that are used to determine whether an action is an outlier, are for both the stops and travels the same: the deviation between the planned and realised duration and the relative deviation.

For the stop actions, all actions were processed in a single pass. This can be done because the stops are all roughly of the same duration, meaning that the deviations and relative deviations can be assumed to come from the same underlying distributions. The MAD method was applied as described in Section 3.2.1, with a threshold values of 3. That is, all data point have a MAD larger than 3 are removed from the data set. Of the 141.809 stop actions, 32.284 were deemed outliers and were removed from the data set.

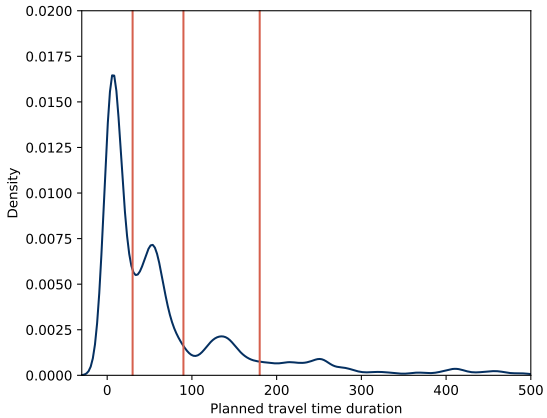

Figure 4.3: KDE plot of the planned duration of travel actions. Red lines indicate the values of the planned duration on which the data was split before outlier removal.

For the travel actions a different approach was used. Since the travel actions can differ much in length, the planned and realised duration are not all of the same scale. To overcome this, we first divide the travel actions in categories based on the planned duration. To determine at what values of the planned duration to split the set, we looked at the distribution of the planned duration. In Figure 4.3, we can see that three clusters can be identified from the Kernel

Density Estimation (KDE) plot of the planned durations. The travel data was split such that these clusters were separated and a group was created for all travel actions longer than the third cluster. These splits were made at 30, 90 and 180 minutes planned duration. For every split, the same cleaning procedure is used as for the stop actions. As there is a relatively large number of short stops, separating these groups before the cleaning ensures that the longer trips can have a larger deviation. When all trips would have been cleaned in a single group, the large number of small trips and their small deltas would cause the estimated distribution of the deviation of all trips to be narrow. This would cause the cleaning to remove the long trips with deltas that are not that large compared to their own duration but that are large when comparing them to the deviation of the short trips.

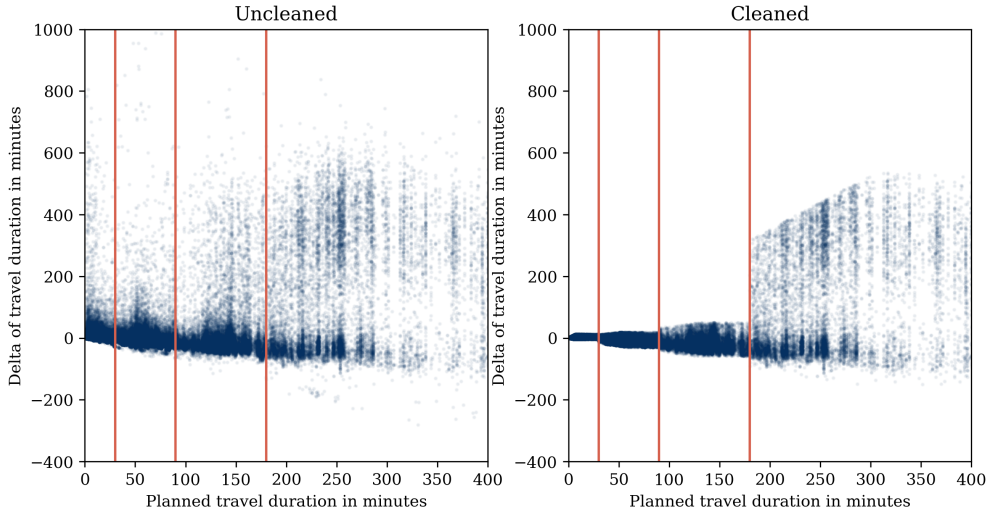

Figure 4.4: Comparison of data set with and without outliers, of the travel actions. Every blue dot represents a data point. Red lines indicate the values at which the data set was split before removing outliers.

In Figure 4.4, the result of removing the outliers can be seen. We can see that the deltas of the cleaned short trips have a narrow distribution. The deltas of the long trips have a more wide distribution. Of the 137.893 travel actions, 21.707 were deemed outliers and have been removed from the data set.

4.3 Distribution estimation

This section will give the methods that are used to estimate the distributions. For both the loading times and the travel times, distributions are needed as input for the simulation model. Next to that, also a distribution is needed to account for some variation in the start time of a trip. For these three sets of distributions, separate models are trained or estimated. Section 4.3.1 describes the steps taken to create a distribution model for the service time. In Section 4.3.2 the process for estimating the distributions for the travel times is described. In Section 4.3.3, this process is described for the distribution of the deviation in start time.

4.3.1 Service time

To create a distribution model for the service times, we need to know on what explanatory variables the service time duration depends. We have some a priori knowledge from the current modelling in ORD and we can extract relations between the explanatory variables and the duration of a stop from the data.

We firstly use the pearson correlation coefficient to gain insight into the variables that influence the realised total duration of a stop. In Figure 4.5, the correlation coefficient matrix can be seen for the data on the stop actions. In this figure, mainly the top row of the matrix is of interest, as this shows the correlation between our dependent variable and the explanatory features. We see that the number of delivered pallets, the planned wait time and the realised time a truck arrives before the start of the time window all have some correlation with the realised total duration of a stop. This indicates that these variables have to be considered when estimating distributions.

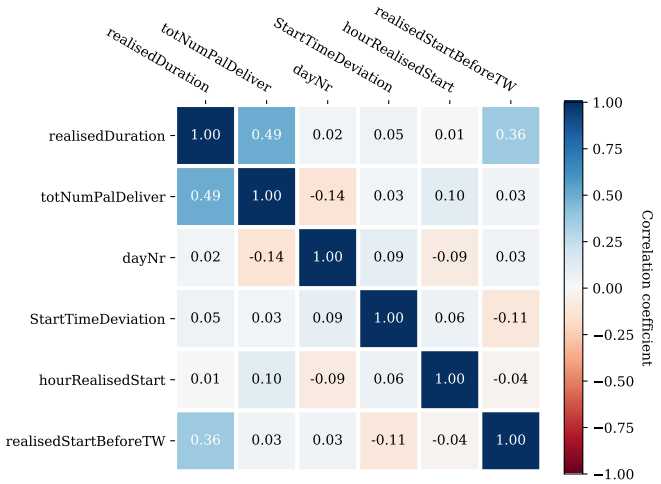

Figure 4.5: Pearson correlation coefficient matrix for the numeric variables of the stop data. The top row shows the correlation of explanatory variables with our independent variable: realised duration.

In Figure 4.6, the features importances of all explanatory variables can be seen. These were calculated by fitting a random forest to the data on the stop actions. This measure gives a rough estimate of which variables influence the duration of a stop. Since the feature importance is not able to deal with correlated features such as $finishAddressID$ and $finishAddressKind$, all variables that have such correlation should still be treated with care. From Figure 4.6 we conclude that the distributions we estimate or learn for the stop time duration will have to deal with the following variables:

- realisedStartBeforeTW
- totNumPalDeliver
- StartTimeDeviation
- finishAddressID

The use of these variables intuitively make sense for the most part. The stop durations that

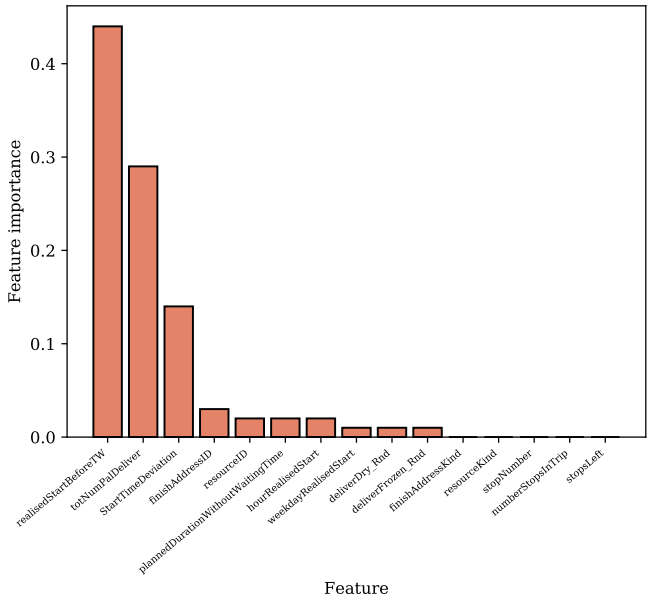

Figure 4.6: Bar chart of feature importance obtained training a random forest using all explanatory variables. Random forest settings: number of trees = 1000, minimum number of samples in leaf = 5.

are in the dataset only give the total time a truck spent at a location, so this will in some cases include waiting time if the truck arrives before the time window. The start before the time window can also be seen as the expected waiting time. Also for different address types separate handling time settings are configured. In these handling time settings there is a fixed and variable time per pallet configured. So one would expect that the amount of pallets and the address at which a stop occurs influence the duration of a stop. The start time deviation of a stop is the only variable that we cannot explain using common sense.

Using the knowledge on which variables influence the total duration of a stop, models for the distribution of the total stop duration will be made. A statistical method will be compared to an advanced statistical method, and the best performing model is chosen. The performance of a model is measured using the cross-validated Average Conditional Log Likelihood (ACLL).

Simple statistical distribution model

For the statistical distribution model we choose to use univariate models. This choice was made since univariate distributions offer a larger set of distribution families to choose from than multivariate distributions. Also, dealing with skewed distributions is more involved in a multivariate setting. When dealing with categorical variables, we have to split the data into multiple categories for both univariate and multivariate distributions. However, smaller amounts of data per category are more problematic in a multivariate setting than in a univariate setting. To estimate distributions of the total stop durations, the data first has to be divided into groups for which we can assume that the observations come from the same distribution. We observed that the duration of a stop is dependent on the location, expected waiting time, deviation in start time and the number of pallets that is delivered. For this distribution model we choose to let the

waiting time and start time deviation out. This means we only use the location and the number of pallets to fit a distribution for the handling time. To account for the location, the data is divided into separate sets for each address kind. While the feature importance of the address is higher than that of the address type, we opt for the simpler model using only the address types to account for the location. Using individual addresses may cause issues since there are some addresses for which there are not enough stops in the data set to estimate a distribution. The amount of pallets that is delivered in a stop is accounted for using a constant time per pallet. This can also be seen as considering the time per pallet to be a constant and attributing all variation in the stop duration to the fixed time. This choice was made based on the fact that the Retailer expected deviations in stop durations to be caused by differences in time it takes to park the truck and sign paperwork.

The time per pallet is estimated using linear regression for each individual address kind. Using the estimated time per pallet we subtract the variable time from the total duration so the remaining duration reflects the estimated fixed time of a stop. The distribution is then fitted for the fixed time of a stop per address kind. The fitting of a distribution on a single group of data points is done by first fitting several different distribution types to the data. The distribution types that are used are:

- Gaussian distribution
- Log-Normal distribution
- Gamma distribution
- Exponential distribution
- Beta distribution
- Inverse Weibull distribution
- Weibull minimum distribution

For each distribution, the maximum likelihood estimation technique is used to estimate the parameters. When all distribution types have been fitted, the Kolmogorov-Smirnov test is used to determine which distribution type has the best fit to the data. The theoretical distribution with the best fit is selected and used afterwards.

Besides the theoretical distributions also empirical distributions are fitted for each of the address kinds. A KDE was fitted by using a rule of thumb to select the bandwidth.

The theoretical and empirical distributions that were fitted on the stop data per address kind can be seen in Figure 4.7.

2-layer statistical distribution model

From the feature importance analysis we know that the location of the stop contains information about the duration. However, the simple statistical model only used the address type and not the individual addresses. Therefore, a statistical model that attempts to find distributions for every individual location is made. However, since for some locations there are little observations present in the data set we need to prune the distributions of locations for which the distribution is worse than the distribution fitted on it's parent address kind. Also a minimal number of data points is set so that no distributions are made if the individual address has less data points than this threshold. So this method will first fit distributions for every address kind as was done

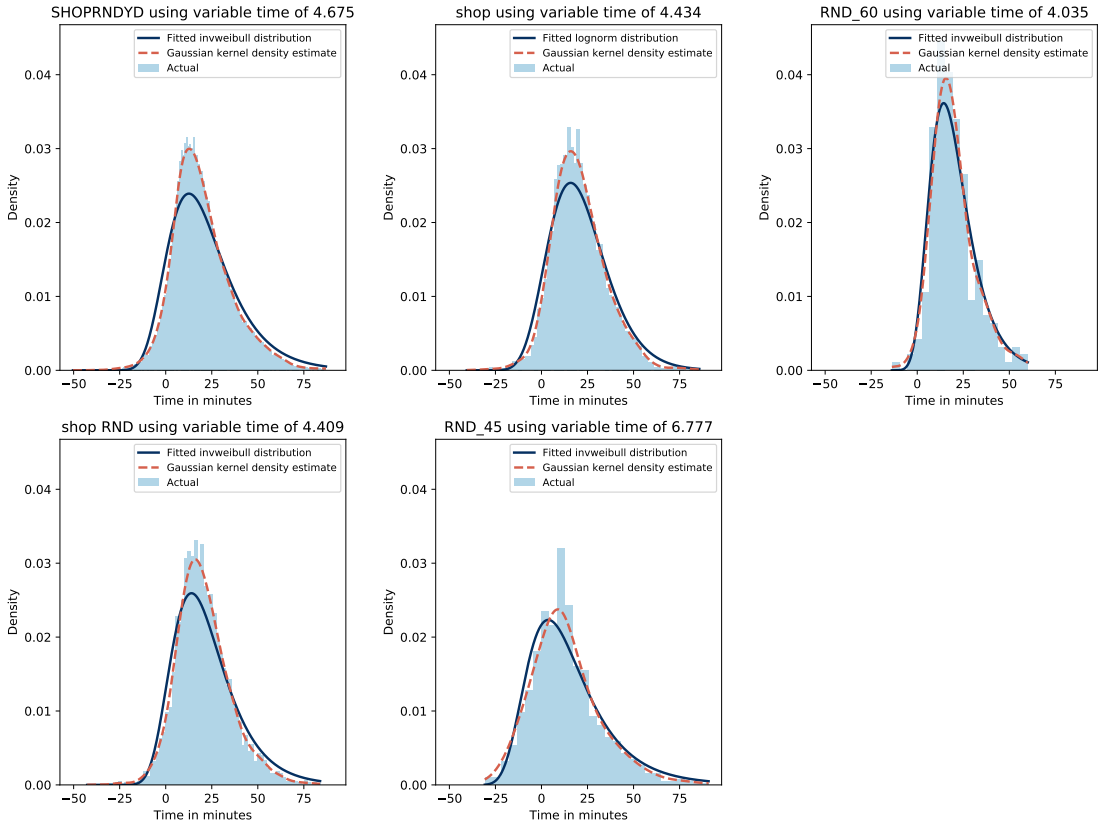

Figure 4.7: Theoretical and empirical distributions fitted to the stop data per addresskind after subtracting the variable time using the estimated time per pallet gained from the regression analysis.

by the previously described method also used for the simple statistical model. Then it will fit for every individual address a distribution and check if the fit of the distribution fitted for the individual address is better than the fit of the distribution that was fitted for the address kind. If the fit is worse, this distribution is removed. This model cannot be visualized easily since it contains over 300 separate distributions. We will therefore simply look at the performance to judge the model.

Table 4.2: Log likelihood for the different models on the test set

Model	**distribution type**	**variable time**	**Mean ACLL**	**Standard deviation ACLL**
Simple statistical	Theoretical	regression estimated	-4.35	0.111
Simple statistical	Empirical	regression estimated	-4.25	0.085
2-layer statistical	Theoretical	regression estimated	-4.26	0.053
2-layer statistical	Empirical	regression estimated	-5.16	0.398

To compare all statistical models, 5-fold cross validation is performed with the ACLL as performance measure. The results can be seen in Table 4.2. From this we can see that the model using an empirical distribution without the distributions on the individual addresses performs slightly better on average than the other models. While it is close to the 2-layer model using theoretical distributions on average, and may be even worse when regarding the standard devia-

tion of the performance, we still prefer the empirical statistical model because it is much simpler. We will use this model to compare to the advanced statistical model that will be fit to select the most suitable model.

Random Forest

To estimate the conditional densities using an advanced statistical method, the Random Forests for Conditional Density Estimation (RFCDE) is used. Before we can estimate the performance of the model, feature selection and hyper parameter tuning has to be done. We choose to first select the features that will be used using some standard hyper parameters. After the feature selection has been done, the hyper-parameters are tuned. The selected features will not be altered during or after the tuning of the hyper-parameters.

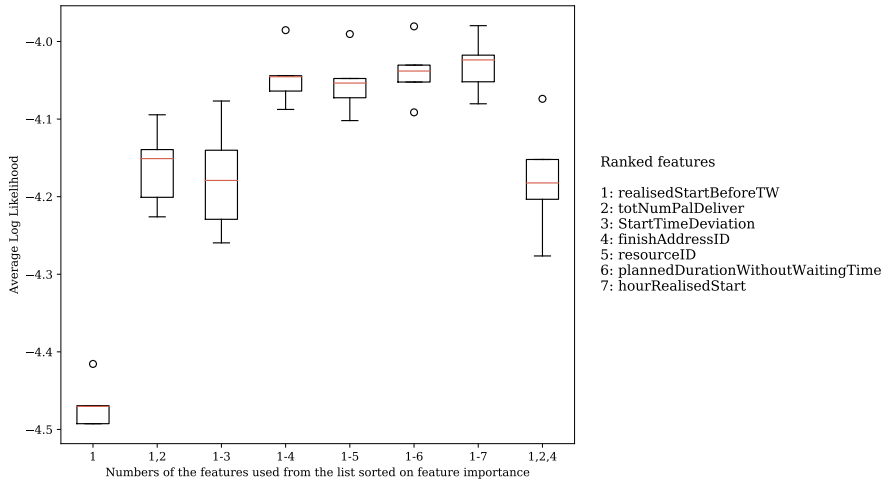

Figure 4.8: 5-fold cross validated log likelihoods of RFCDE trained using different sets of features

Feature selection is done using the list of features sorted on feature importance calculated by the regular RF, as can be found in Figure 4.6. For quick reference we will refer to these features based on their rank, with feature 1 being the most important feature according to the standard RF. The features will be added one by one starting from feature 1. The performance will be measured in the same way as for the statistical methods (5-fold cross validation on ACLL). In Figure 4.8 the performance of the RFs can be seen for different sets of features. In this image some jumps in performance are visible when te 2nd and 4th features are added. After the adding of the 4th feature the performance does not significantly improve. To check whether the addition of the 3rd variable is needed also a forest has been fit on the combination of feature 1,2 and 4. We can see clearly that using the first four features performs better than using features 1,2 and 4. Based on Figure 4.8 we choose to use feature 1 to 4. These are:

- realisedStartBeforeTW
- totNumPalDeliver
- StartTimeDeviation
- finishAddressID

Now the features to train the RFCDE on have been chosen, the hyper-parameters can be

tuned. Using the code package of Pospisil and Lee (2018) there are just three tunable parameters. The number of trees, number of features considered in each split and the minimal number of samples present in a leaf node can be adjusted. Since some of these parameters will also influence the time it takes to draw a random variate of a sample during simulation, the tuning of these parameters will be done considering both the performance and the sample time. Empirical studies have shown that using only a subset of the features for each split has worked well. Suggested rules of thumb are $\lceil \frac{1}{3}k \rceil$ or $\lceil k^{\frac{1}{2}} \rceil$ where k is the total number of features. Since these rules will both result in a value of 2 we decide to simply use this. The tuning of the number of trees and the leaf node size will be done consecutively with the number of trees to be decided and fixed first.

In Figure 4.9 the performance and speed of a RFCDE with differing numbers of trees can be seen. We see that the performance does not increase much after 200 trees while the sample time keeps increasing. We therefore chose to use 200 trees in the RFCDE model. In Figure 4.10 the performance and speed of RFCDE can be seen when using different minimal leaf node sizes. From this figure we first of all see that the performance is not affected much by changes in node size. The sample time does not change much, but it does increase slightly with the node size. We therefore chose to use the default minimal node size of 10.

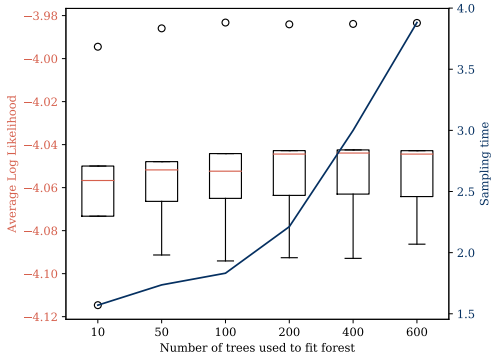

Figure 4.9: Performance (box plot, left axis) and sample time (line, right axis) when using different amounts of trees in the RF. Node size parameters was set to 10.

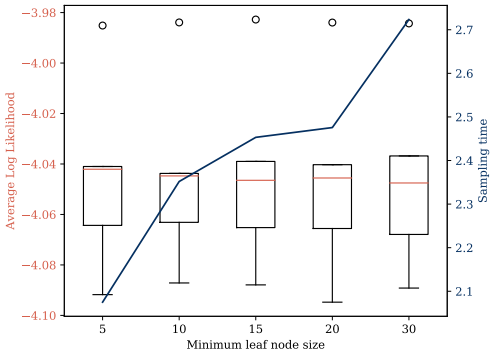

Figure 4.10: Performance (box plot, left axis) and sample time (line, right axis) of RFCDEs using several minimal leaf node sizes.

Conclusion

The RFCDE model has a mean ACLL of -4.04 with a standard deviation of 0.017. The final statistical model has a mean ACLL of -4.25 with a standard deviation of 0.085. From this we conclude that the RFCDE model has a higher overall performance. We therefore chose to use the RFCDE model for the stop time distributions in the simulation.

In order to gain some insight into the RFCDE model, the individual conditional expectation plots of the median of the stop duration distribution are plotted for the three numerical variables in Figure 4.11. These plots portray what would happen to the median of the predicted stop duration distribution, when the feature of interest is altered while the other features remain the same. Every black line in the plot corresponds to one data sample for which this is done. The

red line in the plot is the average of these black lines, also known as the partial dependence. These plots can only show what happen to the median of the distribution, meaning that changes in the spread of a distribution cannot be gathered from these plots. In the left plot in Figure 4.11 the partial dependence of the stop duration on the start time before the time window can be seen. On average the stop duration becomes higher when a truck arrives more minutes before the start of a time window, an effect we would expect. We can also see in this plot that it differs per case how much the stop time duration increases. We also see that the median stop time does not increase any more when arriving more than 80 minutes before the time window. In the middle plot the effect of a larger number of pallets that have to be unloaded in the stop can be seen. As expected, the duration of a stop increases with the number of pallets that have to be unloaded. How much this increases, is somewhat of the same scale for most samples. The right plot show the effect of the deviation in start time from the planned time. A positive value for the start time deviation indicates that the action has started later than originally planned. The plot seems to indicate that on average the duration of a stop tends to be longer when the truck arrives later. However, there seem to be some samples for which this may not be true.

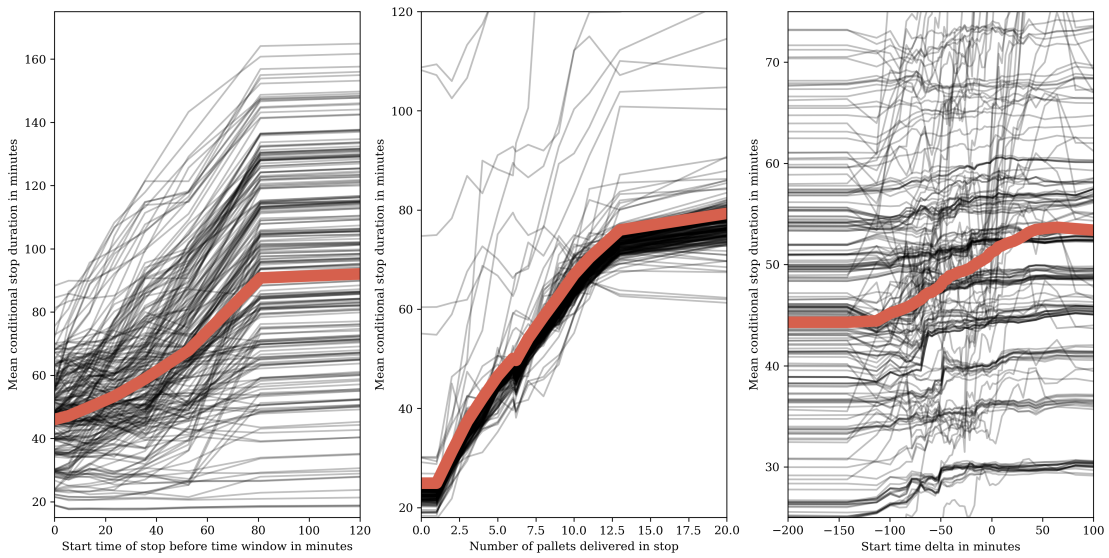

Figure 4.11: Individual conditional expectation plots of the mean stop duration of the three numerical features. The thick red line is the partial dependence, which is the mean of all individual conditional expectation plots.

4.3.2 Travel time

To estimate the duration of a travel action the same procedure will be followed as for the stop durations. In the preliminary data exploration we have seen that on average the travel time predictions made by the map are decent and that the realised duration vary around the predicted duration. Creating distributions for every start and finish pair is not feasible with the amount of data that is present. We therefore chose to create distributions for the deviation between planned and realised durations. During simulation a deviation can be drawn from the distribution and added to the planned duration giving a random travel duration. The deviation in duration is

more on the same scale for different lengths of trips than the total duration. This can be seen as a form of normalization. To chose the best model, we will use a statistical method and an advanced statistical method to estimate the distributions. Before this is done we will first look at what other factors influence the deviation in travel time.

To determine which features influence the deviation in travel duration, we again use a pearson correlation matrix, which can be seen in Figure 4.12. From the correlation matrix, we can see that the planned duration and the travel distance correlate with the deviation in duration. This correlation was already visible in Figure 4.4, where it is clear that the travel actions that have a longer planned duration also deviate more. We can also see that the deviation in duration correlates with the planned waiting time of the next action and the estimated waiting time. The difference between these two variables is small, the former only indicates the waiting time present in the planning, while the latter subtracts the deviation in start time of the travel from this planned waiting time. That this variable shows some correlation can be logically explained. A driver can see in the planning the time windows for the next stop and how much waiting time is planned. Accounting for any previous delays he may have had, subtracting the deviation in start time of the travel action from the planned waiting time at the next stop results in the expected waiting time. When a driver expects to have some waiting time, he can chose to simply take a break during the travel instead of driving through to the next stop and waiting there.

To check the importance of features, again the feature importance is calculated using a standard RF. In Figure 4.13, it is again clearly visible that the estimated waiting time before the next stop is an important feature. After that, the planned duration and distance rank second and third. The other features seem to be relatively unimportant according to the calculations of the RF.

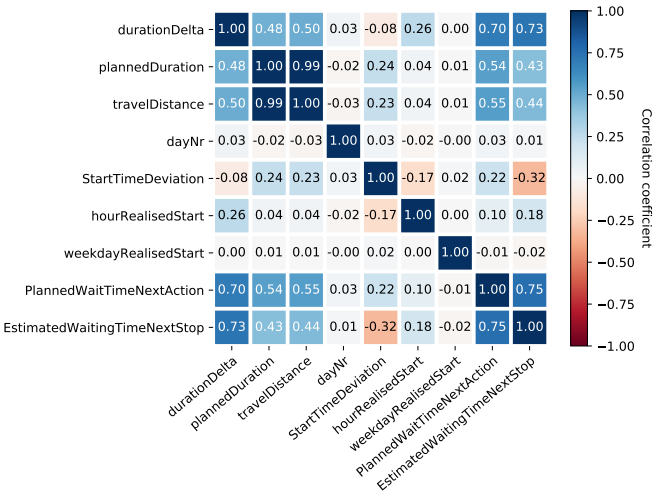

Figure 4.12: Pearson correlation matrix for numerical features of the travel times. durationDelta is the dependent variable.

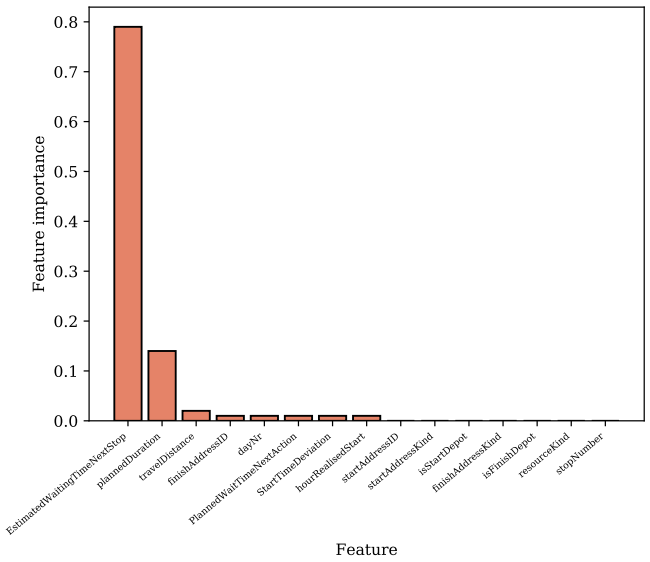

Figure 4.13: Features of the travel times sorted on feature importance as calculated by a RF.

Statistical distribution model

To fit the statistical distribution model we will again use the same approach as was used for the stop duration distributions. From the correlation matrix and feature importance we know that both the estimated waiting time and the planned duration influence the deviation in travel time, meaning that these have to be accounted for somehow in our distribution model.

Table 4.3: The four categories of travel actions with their respective thresholds of the planned total duration in minutes.

Name	**Lower bound**	**Upper bound**
short	0	30
middle-short	30	90
middle-long	90	180
long	180	∞

To account for the planned duration we split up the travel action data based on planned duration. The thresholds for these splits are the same as were used in the cleaning step, these can be found in Table 4.3. The estimated waiting time in the next stop is used to normalize the deviation in travel time. This is done by using a linear regression to estimate the slope of the relation between the estimated waiting time and the travel time deviation for every category. After normalisation, theoretical distributions are fitted in the same way as was done for the stop duration. That is, a set of distribution families is fitted using the maximum likelihood method, from these the best fitting distribution is chosen based on the Kolmogorov-Smirnov test. Besides the theoretical distributions also a empirical distribution is estimated in the form of a KDE. In Figure 4.14, the resulting theoretical and empirical distributions can be seen for the four categories. For every category except the "long"-category, the relation between the estimated waiting time was nearly 0. We therefore decided to not use the normalization on the estimated

waiting time for these categories.

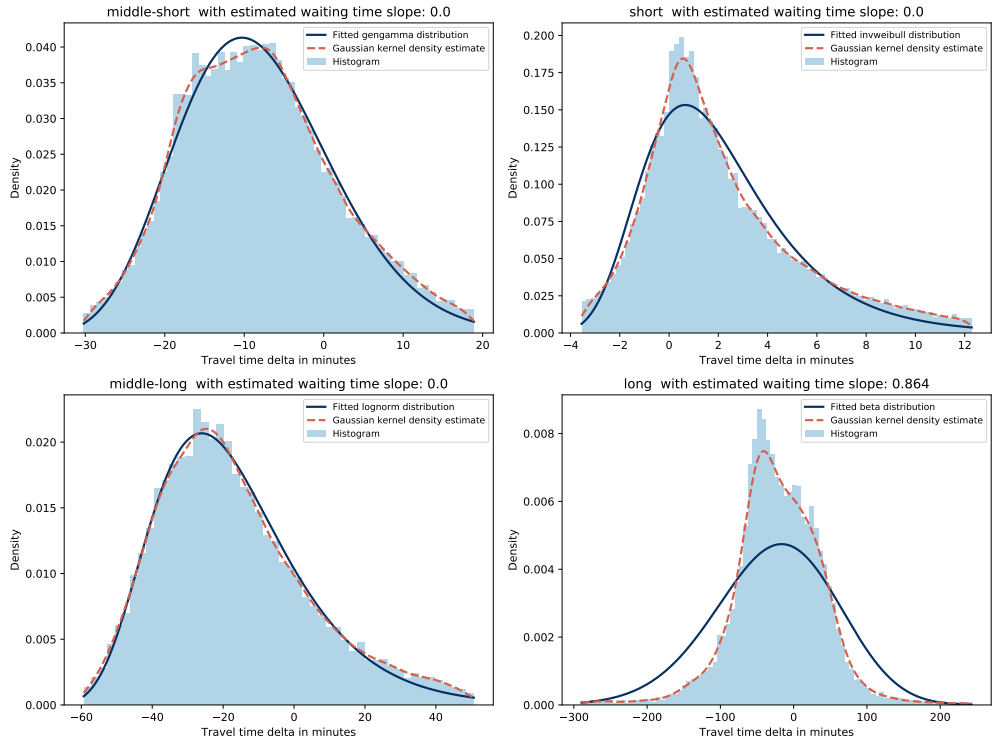

Figure 4.14: Distributions fitted for the travel duration delta on the four categories of travel actions. The distributions are fitted after normalization of the estimated waiting time in the next stop.

In Table 4.4, the performance of the two models can be found. From this we can conclude that the empirical model performs slightly better than the model using theoretical distributions. The empirical model will be compared to the model that is fitted using the advanced statistical method.

Table 4.4: Performance of the two statistical models measured using 5-fold cross validation

Model	**Distribution type**	**Mean ACLL**	**Standard deviation ACLL**
Statistical	Theoretical	-3.52	0.076
Statistical	Empirical	-3.49	0.072

Random Forest

As advanced statistical method a RFCDE is used to estimate densities. The same procedure is followed to select features and tune hyper-parameters, as was done for the density estimation of the stop durations. That means that first the feature selection is done using standard hyper-parameters, after which the hyper-parameters will be tuned one by one.

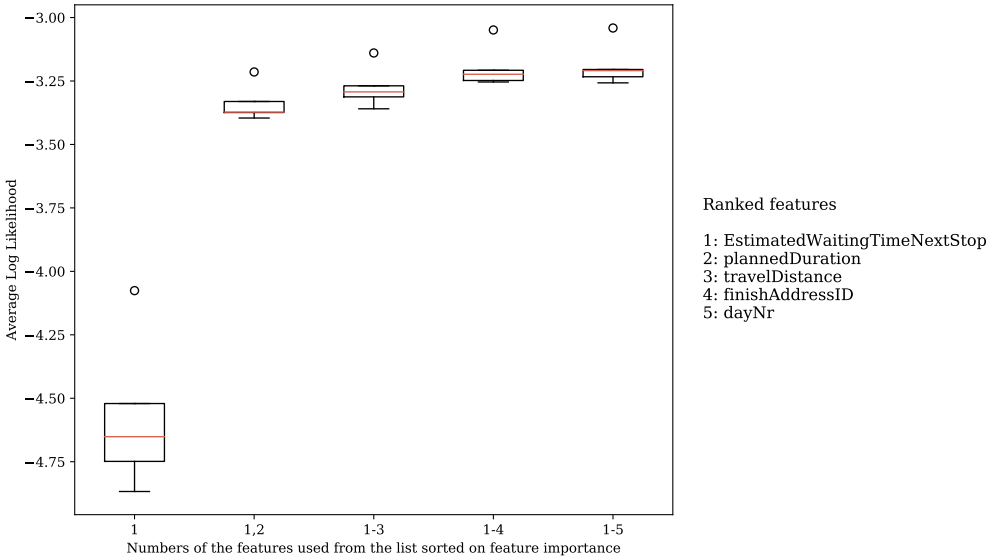

Figure 4.15: Performance of RFCDE trained using different sets of features. Features are added one by one sorted on the feature importance calculated by the vanilla random forest.

Feature selection is done by using the list of features sorted on feature importance, as calculated by the regular RF portrayed in Figure 4.13. These features will be added one by one and the performance of the RFCDE will be measured using 5-fold cross validation. The performance of the resulting RFCDEs can be seen in Figure 4.15. We can see clearly that the addition of the planned duration increases the performance drastically. Adding the distance and finish location features increases the performance slightly. The addition of the 5th feature does not increase the performance noticeably. Therefore, we chose to use the first four features.

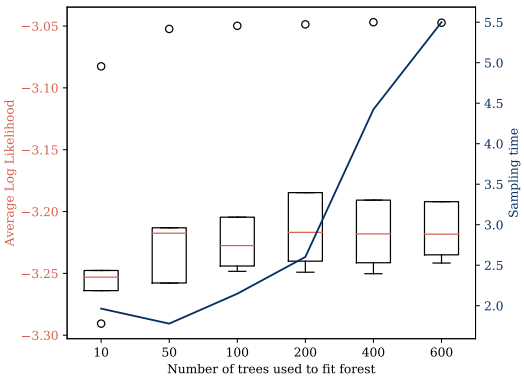

Figure 4.16: The performance (box plot, left axis) and sampling times (line, right axis) when using different amount of trees in the RFCDE.

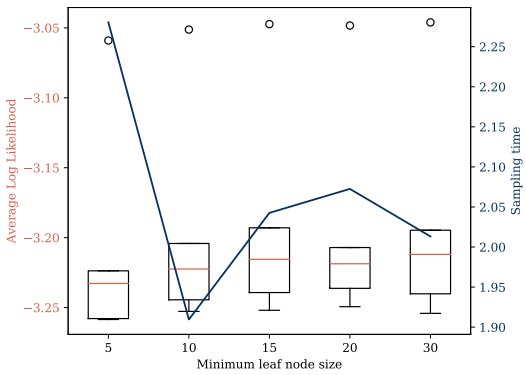

Figure 4.17: Performance (box plot, left axis) and speed (line, right axis) of RFCDE for different leaf node sizes.

The number of features to consider in each split is decided in the same way as was done for the stop durations. This means, that since we have four features, both rules of thumb suggest considering two features in each split. We therefore use a value of two for this hyper-parameter.

In Figure 4.16, the performance and speed can be seen when different amounts of trees are used in the RFCDE. We can see that the performance increases slightly when more trees are used. This increase in performance stagnates between 100 and 200 trees. Also, the time it takes to sample random variates from the model increases with the number of trees. From the amount of 200 trees onwards, the sample time increases almost linearly. We therefore choose to use 200 trees in our model.

In Figure 4.17, performance and speed results of the RFCDE with different numbers of minimal leaf node sizes can be seen. Using a minimal node size of more than 10 does not increase the performance noticeably. Since using an minimal node size of 10 also was the fastest in our test, we chose to use this.

The final RFCDE that will be compared to the statistical model is trained using four features: estimated waiting time, planned duration, distance and finish location. In each split, two features are considered by the model to split on. The forest contains 200 trees, and has a minimal node size of 10.

Conclusion

The statistical model using empirical distributions has a mean ACLL of -3.49 with a standard deviation of 0.072. The final RFCDE model has a mean ACLL of -3.22 with a standard deviation of 0.051. We therefore conclude, that the RFCDE model has the higher overall performance. The RFCDE will be used in the simulation model.

In order to gain some insights into the effects that the features have on the travel time deviation, the individual expectation plots of the mean travel time deviation for the numerical features can be seen in Figure 4.18. In the left most plot we can see that the estimated waiting time increases the mean travel time deviation in some, but not all cases. The same can be said for both the planned duration and the travel distance.

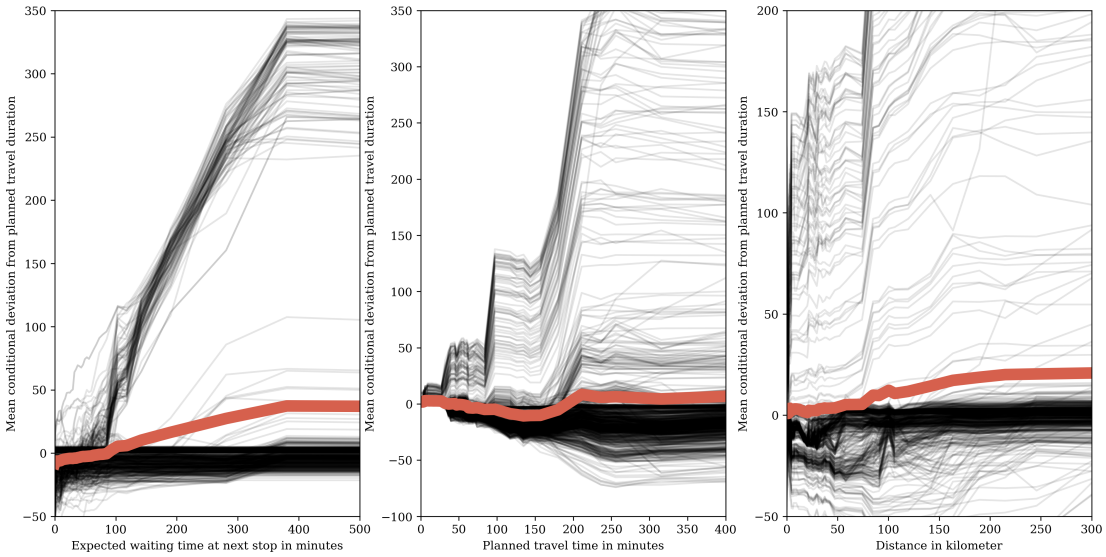

Figure 4.18: Individual conditional expectation plots of the mean stop duration of the three numerical features. The thick red line is the partial dependence, which is the mean of all individual conditional expectation plots.

4.3.3 Trip start time

The start of a trip has some variation with respect to the planned start. To account for this variation in the simulation, a distribution is needed. Since the data on the duration of loading at the depot is not usable, as explained in Section 2.2.4, we cannot use the variation in these loading times as the source of the variation in the departure time from the depot. We may not have valid data on the duration of the stop at the depot, but the departure time from the depot is correct. We therefore will use a distribution that models the deviation in departure time from the depot with respect to the planned departure time, thereby leaving the duration of stops at the depot completely out of the simulation.

To model this variation, we chose a simple univariate distribution that does not depend on any features. Again, both theoretical and statistical distributions have been fit to the departure time deviation. These distributions and the underlying histogram can be seen in Figure 4.19.

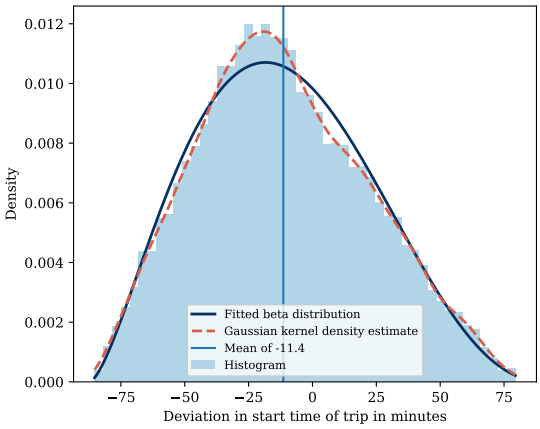

Figure 4.19: Theoretical and empirical distributions fitted on the deviaiton in departure time from the depot.

To test whether the theoretical or empirical distributions would explain unseen data better, 5-fold cross validation is performed using the ACLL as performance measure. In Table 4.5, the results can be seen. Since the theoretical distribution has a lower standard deviation of the ACLL and the means of the two models are similar, we choose to use the theoretical distribution in the simulation.

Table 4.5: Performance of the two statistical models fitted on the departure time deviation measured using 5-fold cross validation.

Distribution type	**Mean ACLL**	**Standard deviation ACLL**
Theoretical	-4.905	0.0036
Empirical	-4.905	0.0041

4.4 Simulation model

This section explains the assumptions that lie at the heart of the simulation model. Then, we describe how the model was implemented. After that, some insight into the validity of the simulation model is given.

4.4.1 Assumptions

To create a simulation model it is essential to list all underlying assumptions. When these assumptions are listed, one can better understand were the discrepancies between the model and the real system lie. Below the assumptions and their explanations can be found.

1. **Routes are not altered during execution**
   The routes that are planned by ORD, are executed as fully as planned. This means that also stops that occur after time windows or opening times are still executed. This assumption prevents us from having to make guesses on what happens when a truck arrives late.
2. **Stop and travel duration are only dependent on the features on which the distribution models are trained**
   This means that the stop duration is dependent only on the location it occurs, the number of pallets that are delivered, the number of minutes of the start before the time window and the deviation in from the planned start time. The duration of a travel action is dependent on the expected waiting time at the next stop, the planned duration, the distance and the destination location. This means that there are no dependencies on for instance drivers or vehicles.
3. **All routes are independent**
   The departure time from the depot varies around the planned time and is independent of all other factors. This means that no dependencies exist between routes that may be part of a larger multi-trip route.
4. **Deliveries are missed when the driver arrives after the time window**
   The Retailer has indicated that both missing time windows and opening times are undesirable, as time windows are always within opening times, we simplify this by considering all deliveries after their time windows to be missed deliveries. We treat the durations of these deliveries as normal and add a penalty for every missed delivery to the total costs.

These assumptions have been checked with the relevant experts at ORTEC. It is possible that assumption 3 could have been avoided when there was valid data available of stops at the depot. Unfortunately, this data is not present and therefore this assumption has to be made.

4.4.2 Model implementation

The simulation model essentially has to calculate the number of deliveries that fall outside their time windows. The input for this model is a planning created by ORD and the distributions models. This was implemented in three parts.

First, the planning file that is created by ORD is converted to a data structure that is more convenient for the simulation. The planning file written by ORD creates separate nodes for every

route. Within a route, all the loading and unloading actions for every order is separated. Also, when a truck has to travel, this travel time is added to the (un)loading action that comes after this travel action. For the simulation input file, this structure is converted so that stop actions and travel actions are separated. Also, all loading actions at the same location are joined so that there is a single action for one stop. This creates a list per route with all the actions that the simulation will sample a duration for. All the features that are needed by the distribution models are written to the actions as attributes.

The second part of the simulation model is tasked with sampling the correct durations from the distribution models. This part is greatly helped by having the right data structure. In a single run of the simulation, we first create a loop that runs over the list of routes. Then, for every route we consecutively draw a duration for every action in this route. This is done for the stop and travel actions by predicting the conditional cumulative distribution based on the attributes of the action from the corresponding RFCDE. We sample a duration by using the inverse transform sampling method. That is, we draw an uniform sample between 0 and 1 and interpret this as a probability. Then, the duration that corresponds to that probability is calculated using the conditional cumulative distribution that was predicted by the RFCDE.

During the last part of a single simulation run, the KPIs are calculated. For every route, the number of stops and corresponding number of pallets that arrive outside the time window are counted. Then the KPIs of the case are calculated by summing the KPIs of the individual routes. When multiple cases are used, there is an additional summation that joins the KPIs of the cases.

4.4.3 Model validation

To validate the simulation model, a comparison will be made between the historic data and the simulation model. In order to make a good comparison, we will use the routes from the database and compare their respective realisations to realised routes created by the simulation model when the same planned routes are given as input. This means that the input for this simulation does not come from ORD, but was directly created from the database.

To achieve this, a script was written to create input files for the simulation using planned routes from the data set. However, since the cleaning of data was done on an action level, not all routes could be used, as some actions may have been removed from the route in the data set. Therefore, we used routes for which none of the actions were removed from the data set to create a set of "clean" routes. From this set of routes, which contained over 32000 routes, 500 routes were picked randomly to create the validation set. This was done to reduce the computational time. During the simulation of the validation set, the results of every action were written to a dataframe so that these can be compared to the historic durations. This increased the computation time for simulating 500 routes from approximately 5 seconds to 10 minutes.

The simulation model performed 5 runs using the validation set as input. To compare the results of the simulation to the realised routes in the data set, the difference between the realised historic durations and simulated realisations of the same actions or routes are investigated.

In Figure 4.20, the deviation in minutes between the historic realised times and the simulated realised times can be seen. The deviations have been calculated for stop actions, travel actions

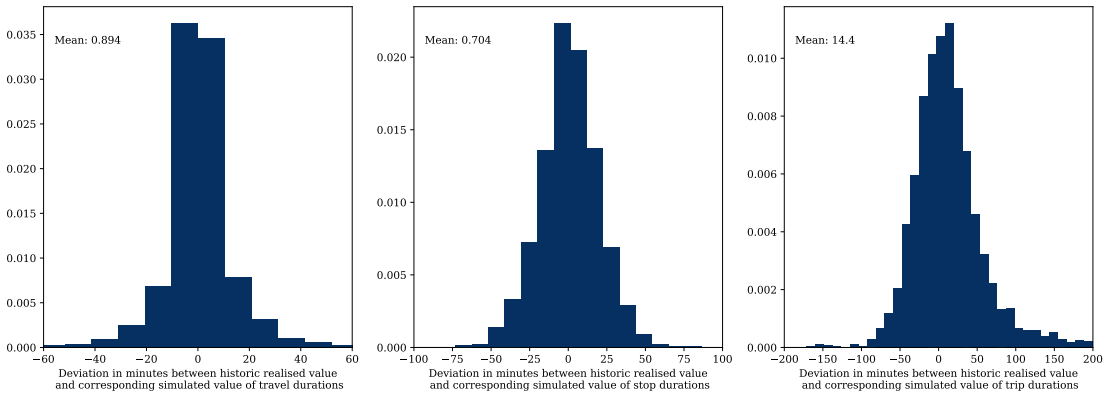

Figure 4.20: Deviation between the realised durations in the historic dataset and the realised durations of the simulation when the same planned routes are used.

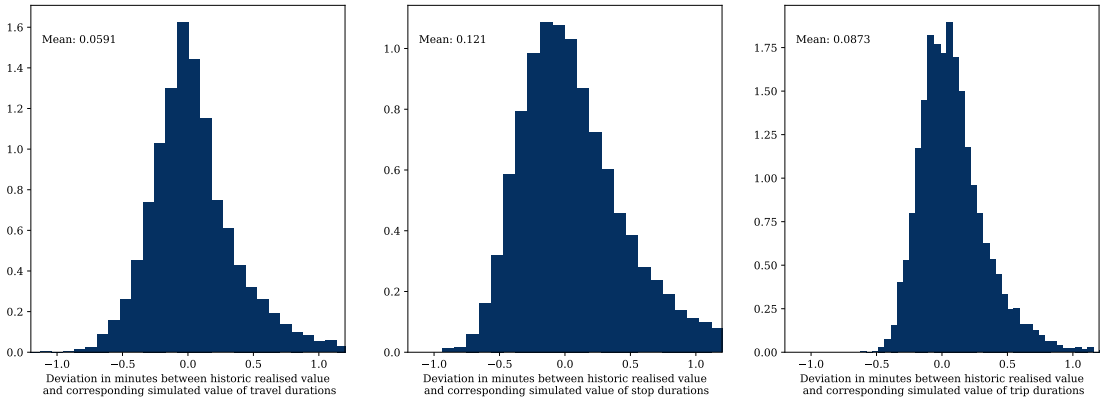

Figure 4.21: Relative deviation between the realised durations in the historic dataset and the realised durations of the simulation when the same planned routes are used.

and complete routes. We can see that both the simulated stop and travel durations are slightly longer than the historic durations. The full route length is almost 15 minutes longer on average than the historic realised route duration. This means that our simulation somewhat overestimates the durations.

In Figure 4.21, the relative deviations between the realised historic durations and simulated realisations is shown. These deviations are relative to the realised historic durations. Here, not to our surprise, we also see that our simulation slightly overestimates the durations compared to the historic durations. However, we can now see that the overestimation might be caused by the stop actions, as these relatively deviate on average more than the travel actions or routes.

These durations are not the KPIs that will be used during the optimization. Since our service costs are determined by a number of missed time windows, these should be investigated as well. This was done by looking at the percentage of stops that started after the time window. In the historic realised routes from the validation set, 8.34% of the stops started after the time window. In the simulated realisations this was 8.97% of the stops on average. Since this model will be used only to compare KPIs of different input parameter settings with each other and not with the KPIs of the real system, this increase is small enough for our model to be used.

4.4.4 Conclusion

In this section the assumptions made by the simulation model have been explained. Also, a validation of the simulation model has been performed by comparing it to routes from the data set. The validation of the simulation model has revealed that it slightly overestimates some durations. This could be caused by some missing dependencies or assumptions that are violated. However, we believe that the model could not be improved with the data that we have used. Also, the percentage of stops that are outside the time windows is overestimated by 7.6% by the simulation model. Besides this, the simulation model will be used to judge the effect of changing input data values. This effect will be measured by comparing outputs of the simulation model with new settings to the output of the simulation with the old setting. So, while the model may not give the most reliable absolute values for some KPIs, we can still use it to investigate the relative effect of changing parameters.

Chapter 5

Simulation Optimization

This chapter explains how a sequential optimization algorithm can be used to find the best input data values. Section 5.1 explains a method that can be used to incorporate knowledge on the input data into the optimization. In Section 5.2, the algorithm that was used is described. Section 5.3 gives some results of running the algorithm. In Section 5.4, proposed new input data sets are compared to each other and the current setting. Finally, Section 5.5 draws conclusions from the results seen in this chapter.

5.1 Reparametrization

Optimizing the input of a simulation increases in difficulty with every added parameter to optimize over. Therefore, it is useful to reduce the amount of parameters that need to be optimize. This will reduce the number of iterations the optimization algorithm needs to find the location of the minimum. Given the relatively long computation time per iteration, this dimensionality reduction is needed to keep the total computation time in check. We use the data set to create a model for the input data values to reduce the number of parameters to optimize over to 1.

The motivation behind this model starts with realizing that when the average parameters are configured, approximately 50% of the stops will be shorter than planned and 50% will be longer. Since missing time windows will incur some costs, we might want to slightly overestimate these stop times in the planning so that a larger percentage of the stops are completed within their planned durations. We call this percentage the *service level*. So, to reduce the number of parameters to optimize over, we use this service level instead of the individual handling times setting for all five address types. Thus, we need a model that gives us the handling time settings that correspond to any percentage of stops that ought to be completed within their planned time. The handling time settings generated by this model will not guarantee that a certain stop service level is reached. We therefore call this estimated percentage of stops the *handling time percentile* to differentiate it with the service level. The reason for this difference between the estimated and realised percentage will be explained later. When this model is used in the simulation optimization, the optimization algorithm will attempt to find the handling time percentile that gives the lowest total costs. Thus, in every iteration, the optimization algorithm proposes a percentage which is then converted to the handling time settings for every address type by our

model which are then used to plan all deliveries.

An added benefit of this dimensional reduction is that all address types will be treated approximately equal. When we would have used the separate handling times to optimize over, the effect of changing handling times of the address types that have only a limited number of orders, is hard to judge and therefore difficult to find the best value for. The output of the simulation would then be dominated by the address types that have relatively many orders.

To find the handling time setting that correspond to the handling time percentiles, we use an RFCDE model. Recall that the model that is used for the distributions of the stop times in the simulation model was trained on four features: number of delivered pallets, address, realised start before time window and the deviation in start time from planning. The model that will be used to calculate the distribution of the handling times is trained using these same features except using the address type instead of the individual address. The set of hyper-parameters that is used to train the handling time distribution model is the same as the set that was used for the stop time distributions.

This RFCDE-model predicts a cumulative distribution for every combination of the four features. We use these predicted cumulative distributions to calculate what the handling times setting should be for a certain handling time percentile. Since in the planning ORD explicitly plans waiting time, we want to configure the handling times so that these do not include this waiting time. Also, in the planning we assume that all actions start at their planned time. For these two reasons, we use the RFCDE-model to predict the distributions of the handling times when setting both the start time before time window and the deviation from the planned start time to 0. This reduces our RFCDE-model to accept only the address type and number of pallets and returns a conditional cumulative distribution. The cumulative distribution is the relation between the duration of the stop and the percentile of realised stops that are shorter than that duration. To calculate the relation between the handling time percentile and the individual handling time settings, we use a linear regression to determine the fixed and variable handling time. This is done per address type and percentile. So, for a certain address type and percentile, the RFCDE-model predicts the distribution of the stop duration on that address type for a range of pallets amounts, and takes the duration from these distributions that correspond to the desired percentile. Then a linear regression is performed on this set of durations and number of pallets to find the fixed and variable handling time for this address type and percentile.

The resulting relation between the handling time percentile and the corresponding handling times settings can be found in Figure 5.1. These have been plotted for percentiles between 0.01 and 0.99 to prevent the figure from being distorted from strange durations at the limits. The axes of all plots are aligned equally so that the differences between the address types can more easily be spotted. We notice that the relation between the input data values and the handling time percentile seems to be almost linear between a percentile of 0.2 and 0.7. A higher handling time percentile than 0.7, is associated with a steep rise of the fixed handling time for all address types. Note that the estimations of the parameters for the address types “RND_45” and “RND_60” have a larger uncertainty. This can be explained by the fact that these address types had fewer associated addresses and therefore fewer data points than the other address types. When looking at the fixed time, we note that the address types “SHOPRNDYD” and “shop” have a slightly higher fixed time in general than the other address types. When looking at the

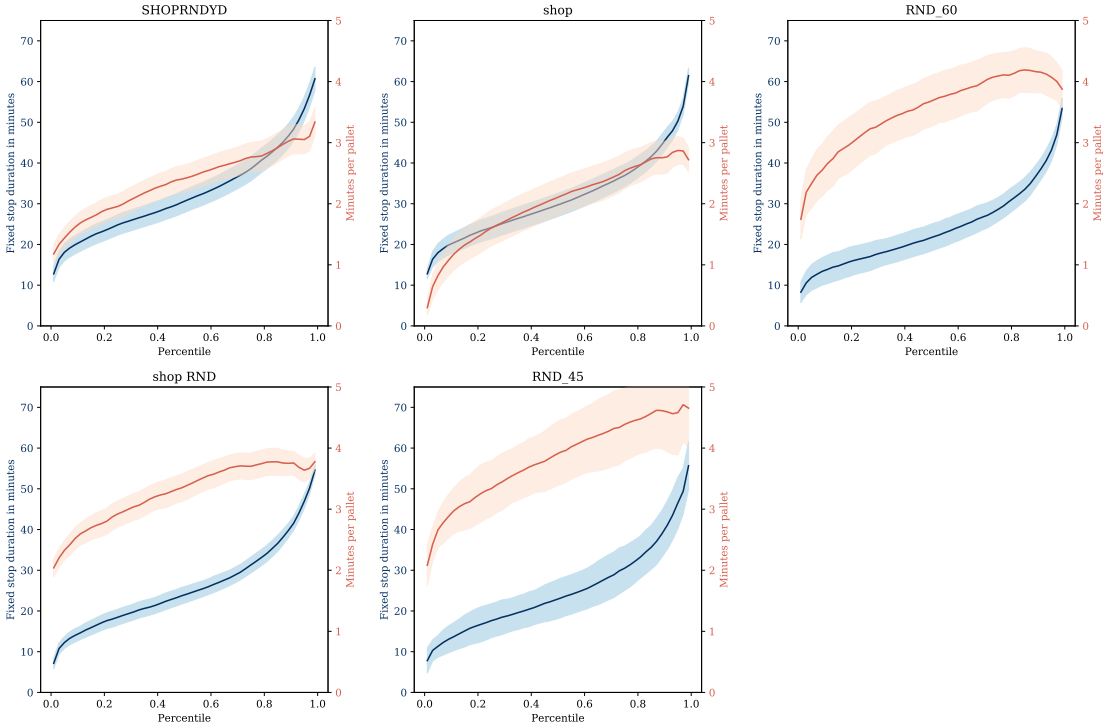

Figure 5.1: Estimated parameters per address type for every handling time percentile. The blue line and left axis in each plot give the fixed handling time. The red line and right axis gives the variable time per pallet of the handling time setting. The percentiles range between 0.01 and 0.99.

variable time however, the address types “SHOPRNDYD” and “shop” have a lower associated time per pallet than the other address types.

The handling time percentile that we use looks a lot like the service level of the stop. However, during the simulation it is not necessarily the case that the handling time setting corresponding to a certain handling time percentile will achieve that service level for the stops. This is caused by the fact that during the simulation also the waiting time is included in the stop duration, as was the case in the data set. Therefore, we simply do not know how long the unloading actually took and how long the driver was waiting. Also, the deviation from the planned start time is taken into account in the simulation while we purposefully filtered that out of the handling time percentile model.

We would like to point out that the creation of such a model, that relates a handling time percentile for the stops to an estimated setting, is not restricted to the RFCDE model that we have used. The statistical model that was fitted to the stop data could also have been used to create such a model. If that would have been done however, the variable time would simply be a fixed parameter for all address types, as the statistical model assumed that all variation in duration comes from the fixed handling time. We also could have used the exact same model that was used as distribution model for the stops, with the address as feature and not the address types. That model would then be able to give a relation between handling time settings that should be set at every individual address and the service level. Averaging these over all

addresses of a certain address type would then also give the relation between the service level and address types. This method is slightly different from the method that we have chosen. If first all handling times setting models for all individual addresses are calculated and then averaged to find the handling time setting for the address type, the durations at every address are weighed equally. The method that we have opted for, implicitly favours addresses that occur more often in the data. We think however, that it is useful to calculate the settings of the address type by favouring addresses more often in the data set as these are more often visited and therefore should be considered more than the addresses that are not visited as much.

5.2 Optimization set up

To find a good set of handling times settings, we use an optimization algorithm. This optimization algorithm will use the model explained in Section 5.1 to create handling time settings corresponding to handling time percentiles in order to reduce the dimensionality of the optimization problem. This means that the optimization algorithm has a one-dimensional and continuous search space.

5.2.1 KPI

To find the best value in this one-dimensional handling time percentile search space, the performance of a setting has to be measured using a KPI. As mentioned in Section 2.2.5, the total costs of a set of routes is determined by the plan costs and the number deliveries that happen after the time window. To get a KPI that reflects total costs, the deliveries that happen after their time window have to be converted to costs. We do this by using the plan costs per pallet and multiplying this by the total number of pallets that were delivered after their time window as a rough estimation of the costs that would be made if the pallets had to be redelivered. To control how undesirable it is to miss deliveries, a factor is used to multiply these service costs by. The calculation of this KPI is then given by

$$TC = PC + SCF \cdot MP \cdot \frac{PC}{TP}$$

where TC is the total cost, PC the plan costs, SCF the service cost factor, MP the number of missed pallets and TP the total number of pallets. This KPI balances the importance of finding a tight schedule with the value of delivering pallets within their time window. How important one aspect is with respect to the other can be adjusted using the service cost factor. To create an easily interpretable KPI for which we can quickly see how well it performs compared to the current situation, the total costs are always displayed as a fraction of the total costs that the handling times settings that are currently configured gives in the same situation. With same situation here we mean, the same cases, service cost factor and simulation runs. This value will be equal to 1 if the costs of the new handling time setting are the same as the current costs, and it will be below 1 if the costs of the new settings are lower.

5.2.2 Cases

In every iteration of our optimizer, routes are first planned by ORD and realisations are simulated by our simulation model to calculate the total costs. To do this, some sets of orders and vehicles are needed that represent situations at the Retailer, so that these can be planned by ORD. The obvious choice here is to retrieve all data from the data set on a particular day and create a case file with all orders that were planned on that day with all vehicles that were used. So, we define a case as the historical instance of a day as found in the data set. This was implemented by creating a script that is able to create input files for ORD based on the data set. As a consequence, only the orders present in the cleaned data set have been used to create the case files. Optimizing the performance of a handling time setting on a single case may lead to overfitting on that case. Therefore a set of cases is used during the simulation optimization process. The more cases used in every iteration, the better the estimation of the general performance of a handling time setting. However, planning a single case takes 15 minutes on average. We therefore choose to use four cases that are planned and simulated in every iteration. These four cases have been chosen so that they all occurred on different days of the week and spread over the year of data that was available. Taking the cases spread over the year automatically caused some cases to be larger than others, as the demand fluctuates seasonally at the Retailer.

5.2.3 Algorithm

ORD, the simulation model and the handling time model combined are seen as a large and expensive black box function that gives a single output value when evaluating a single input value by the optimization algorithm. A depiction of how these models are combined and what their inputs and outputs are can be seen in Figure 5.2. The output of this black box function contains noise, and no derivatives are available. When we use the term input with respect to the optimization algorithm, a handling time percentile is meant. This also applies to the output of our function to optimize, as it is the total cost.

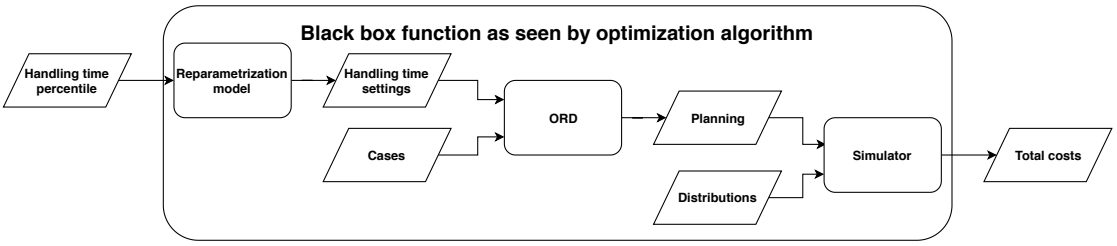

Figure 5.2: Schematics of the different modules and their respective inputs and outputs of the black box function that is optimized.

As explained in Section 3.4.3, to sequentially decide which percentile of the handling times settings to test, we use a Bayesian Optimization algorithm. This means that we use a Gaussian Process surrogate model to estimate the relation between our input and outputs during optimization. The optimization algorithm was implemented using the Scikit-Optimize package in Python. The Expected Improvement is used as acquisition function to decide which input value to evaluate next. That is, the input value that has the highest Expected Improvement is evaluate

in the next iteration. This acquisition function has no parameters that need to be tuned. The use of this acquisition function automatically balances exploration and exploitation in the learning process. An example of the acquisition function can be seen in Figure 5.3. Here, we see how the GP approximates the relation between inputs and outputs. We also see that the GP is uncertain about a section of the search space. The red line in Figure 5.3 indicates the EI-function. We can see that it measures a point close to the current minimum next. However, another section of the input search space is also interesting as there is still some uncertainty there. This corresponds to the trade-off between exploration of uncertain areas and the exploitation of the region where the model suspects the minimum to be.

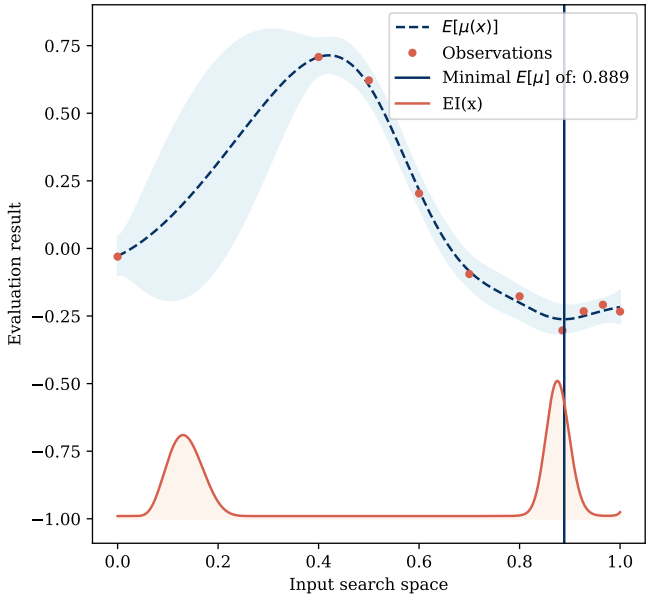

Figure 5.3: Example optimization problem showing how the GP approximation of the input output relation behaves. The blue shaded area indicates the uncertainty of the GP-model. The red line portrays the Expected Improvement acquisition function, this been enlarged and shifted to provide a clear image.

To initialize the GP, some input-output value pairs are needed. These are typically determined in one of two ways. One way is to uniformly sample a set of input values across the input search space and evaluate all these points. Another way is to determine the input values using a priori knowledge on the function. After the initialization set of input values have been evaluated, the acquisition function sequentially determines the following input values. To initialize the optimization algorithm we have chosen the second option. We suspect that the handling time percentile will have to lie somewhere between 0.4 and 0.8, and therefore initialize the model with a range of values between 0.4 and 0.8 with increments of 0.1. This range was chosen based on expert opinion.

When the output values of these four initial input values have been calculated, the GP-model is fitted to the set of input-output pairs. Then, the point on the input space that maximizes the acquisition function is calculated using a multi-start gradient ascent optimization algorithm. This input space is restricted to the range 0.01 to 0.99 to prevent the measurement of strange

handling time settings at the limits. This input percentile is then evaluated by calculating the corresponding handling times using the model from Section 5.1, optimizing the planning of all the cases using ORD, simulating the planned routes and calculating the total costs of the planning and simulation. The evaluated point and corresponding output are then appended to the list of observations and the GP-model is updated. This process is then performed iteratively.

The optimization algorithm also requires a stopping criterium. Typically, this is either a maximum number of iterations, or a criterion that stops the algorithm when the minimal distance between the next input value and all previous measured input values gets below a certain threshold. We opt here for the simpler variant using a maximal number of iterations. This can be configured more easily, only requiring knowledge on the computational duration of an iteration and the total computation time that is available. We chose to use a maximum of 15 iterations for every optimization run without any early stopping rules.

5.3 Experiments

Since the absolute KPIs of our simulation are not relevant, we first run the simulation with the handling times that are currently configured at the Retailer, so that we later can calculate the KPIs of our new handling time settings relative to the current situation.

The simulation using the current handling time configuration was also used to determine the number of simulation runs that should be performed for every case in every iteration. The heuristic used by ORD is fully deterministic, meaning that given the same case and handling time settings, the planning that is created will be the same every time. Therefore the KPIs that are determined by the planning, like plan costs or total distance, cannot be used to determine how many simulations have to be performed, as these have no variation at all. We used the total number of pallets delivered after their time window as KPI to determine the number of replications. We first replicated the simulation 1000 times using the current handling time configuration. This resulted in on average 41.2 pallets arriving after the time window of a total of 6388 pallets. The standard deviation of this KPI was 18.5. We first of all note that the percentage of pallets that arrive late (0.29%) seems quite low. This could be caused by some assumptions that have to be made to create the simulation model. For instance, the fact that in our simulation all trips are independent, may cause the number of late pallets to drop, as the deviation of earlier trips by the same truck do not cause later trips by the same truck to be late.

Besides the low average number of late pallets, the standard deviation is relatively high compared to the average number of late pallet indicating that it fluctuates a lot in every replication. Typically, the number of replications is determined by checking if the relative confidence interval of the mean is below a certain threshold. This calculation was performed with a confidence level of 0.05 and a relative width of the confidence interval of 0.05 giving a value around 350 replications. However, we have chosen to use slightly fewer replications during the iterations of the optimization to lower the computation time. We use 200 replications in every iteration. The computation time for planning a single case is around 15 minutes. Replicating the simulation for a single case 200 times also takes roughly 15 minutes. Some computational efficiency is gained through the fact that ORD can optimize the cases in parallel in different processes. This reduces the computation time for planning and replicating the simulation 200 times for 4 cases

from approximately 120 minutes to around 90 minutes. This parallelization was not possible for the simulation, as the package used for the distribution models did not allow for copying of the models. This is needed to run the simulations fully in parallel using separate parts of the memory.

In the remainder of this section, the results of the experiments are shown. In Section 5.3.1, the first experiment with a cost factor of 1 is given. Section 5.3.2 describes the experiment where different cost factors are used to show that this causes the resulting minimum to shift. Finally, to show the convergence and functioning of the algorithm, a cost factor is estimated and subsequently used in Section 5.3.3.

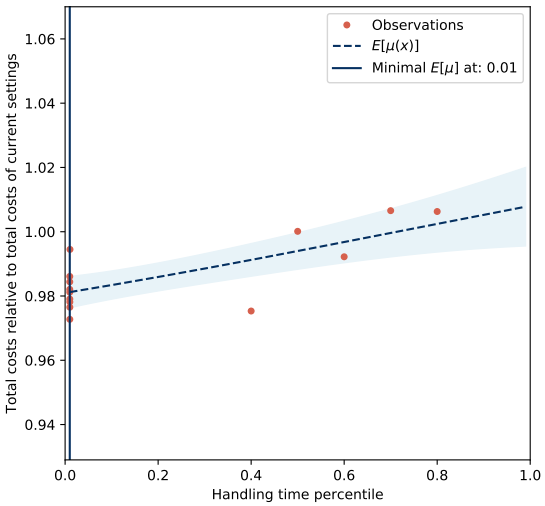

Figure 5.4: Estimated relation between handling time percentile and total costs after 15 iterations with a service cost factor of 1. Total costs are relative to the total costs of the current handling time setting.

5.3.1 Initial experiment

We first run the optimization algorithm with a cost factor of 1. This cost factor was chosen as it is the most logical factor, simply stating that for every missed pallet a penalty costs equalling the average plan costs of a pallet is charged. In Figure 5.4, the results can be seen of 15 iterations of the optimization algorithm with a service cost factor of 1. We can see that the optimizer has not found a balanced setting and simply minimized the handling times. This is supported by the fact that a large portion of the measurements is performed close to the handling time percentile 0.01. This seems to have such a large effect on the planning costs that the increased number of pallets that arrive late are of no effect. Using a handling time percentile of 0.01 increases the number of late pallets to an average of 122.6. However, this is still little compared to the total number of pallets that are transported, which makes the effects of the number of pallets that are late on the total costs small. This result is not desirable for the Retailer, as a 200% increase in late arrivals does not weigh up to a 2% decrease in planning costs. This means that we have to find another way to get a balanced handling time setting.

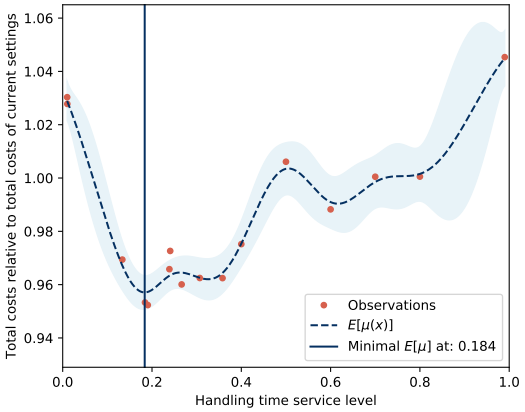

Figure 5.5: Estimated relation between handling time percentile and total costs after 15 iterations with a service cost factor of 10.

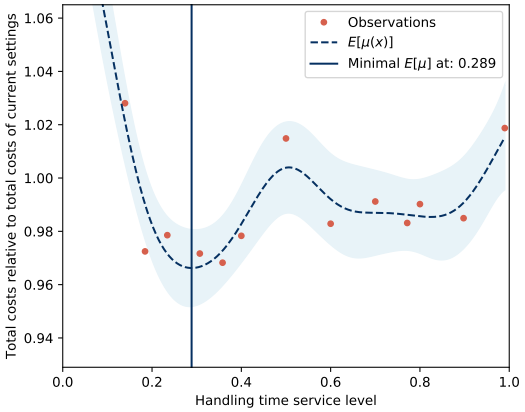

Figure 5.6: Estimated relation between handling time percentile and total costs after 15 iterations with a service cost factor of 20.

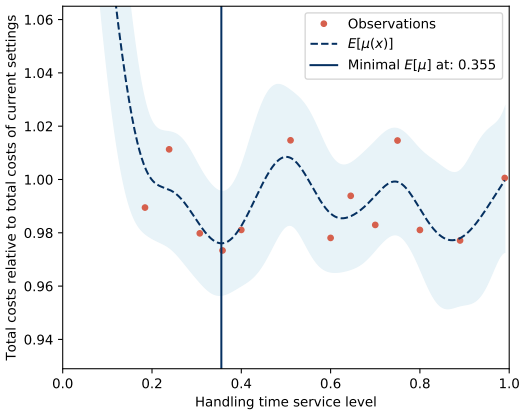

Figure 5.7: Estimated relation between handling time percentile and total costs after 15 iterations with a service cost factor of 30.

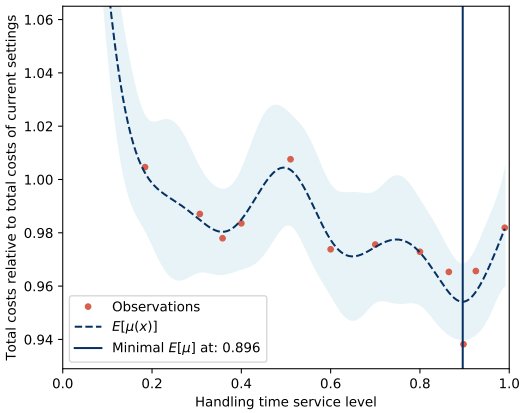

Figure 5.8: Estimated relation between handling time percentile and total costs after 15 iterations with a service cost factor of 40.

5.3.2 Different cost factors

That this cost factor of 1 does not give a balanced result may be caused by some of the assumptions underlying the simulation model. We will show that the optimization method works when using a total cost KPI that balances the plan costs and the time window violations. We do this by simply increasing the service cost factor, so that late pallets are more severely penalized. We have run the simulation optimization algorithm using service costs factors of 10, 20, 30 and 40. We can use the insights that are gained from these runs, but not the optimal settings that come out of them, as then we would be looking for the service cost factor that would give the most logical result. This would make us tune the method until the result is to our liking, instead of choosing a service cost factor that can be explained.

In Figures 5.5 to 5.8, the results can be seen for the four different service costs factors. Comparing the handling time percentiles that minimize the total costs for the different service cost factors, reveals that the handling time percentile that minimizes the total costs increases

as the time window violations cost more. However, we also see there is a lot of noise in the observations. Further inspection revealed that this noise in the measurements does not only come from the service costs that are calculated by the simulation, but also from the plan costs calculated by ORD. For instance, the planning that was made using a handling time percentile of 0.6 has overall lower planning costs than the planning that was made at a handling time percentile of 0.5, indicating that a solution has been found with less distance or lower total duration. This better planning should also be possible with the handling time settings of a 0.5 percentile, as the stops only get shorter. The shorter stops may cause an increase in waiting time, but the total duration should not increase overall. This variation in solution quality can be attributed to the heuristic nature of ORD.

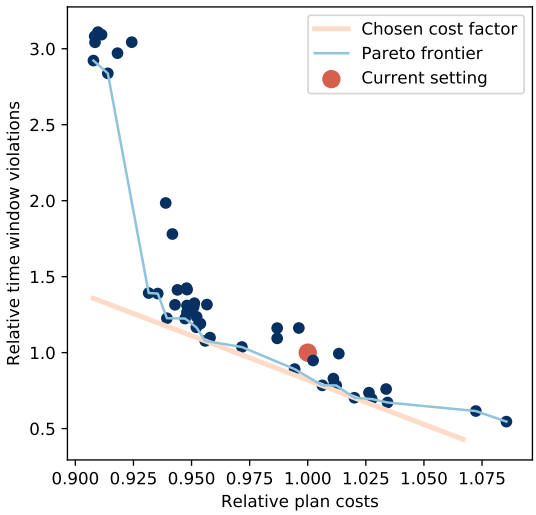

Figure 5.9: Relative plan costs and relative time window violations of all measurements made during the optimization using the service cost factors 10, 20, 30 and 40. The Pareto frontier indicates the measurements that are dominating. The red line portrays the position of solutions with equal total costs when the service cost factor is set to 28.57

5.3.3 Estimated cost factor

Unfortunately, the handling time percentiles that were found in Figures 5.5 to 5.8 are not the percentiles that would yield the lowest costs at the Retailer as the service cost factors were chosen arbitrarily. Discussions with the Retailer have not resulted in a service cost factor that would be appropriate. Therefore, to show that the algorithm is able to find a good setting, we will make an educated guess of the service cost factor based on the previous measurements and use this to run the optimization. We guess the service cost factor based on the Pareto frontier shown in Figure 5.9. In this figure the two KPIs of interest are on the x- and y-axis. We can clearly see the trade-off between the plan costs and time window violations in this figure. Choosing a service cost factor is equivalent to choosing the slope of a line indicating solutions of equal value in the Pareto plot. It is choosing how important we value one KPI compared to the other KPI. If we then find the line with this slope that has the lowest intercept while still passing through a point in the Pareto plot, we find the point that is optimal for that service cost factor. The distance

from a point to the line indicated how much worse that measurement is to the optimal. Since we believe the current configuration is decent, we have chosen a service cost factor that minimizes the distance from the current configuration to the optimal found solution. This is the red line that can be seen in Figure 5.9. This corresponds to choosing a service cost factor of 28.57.

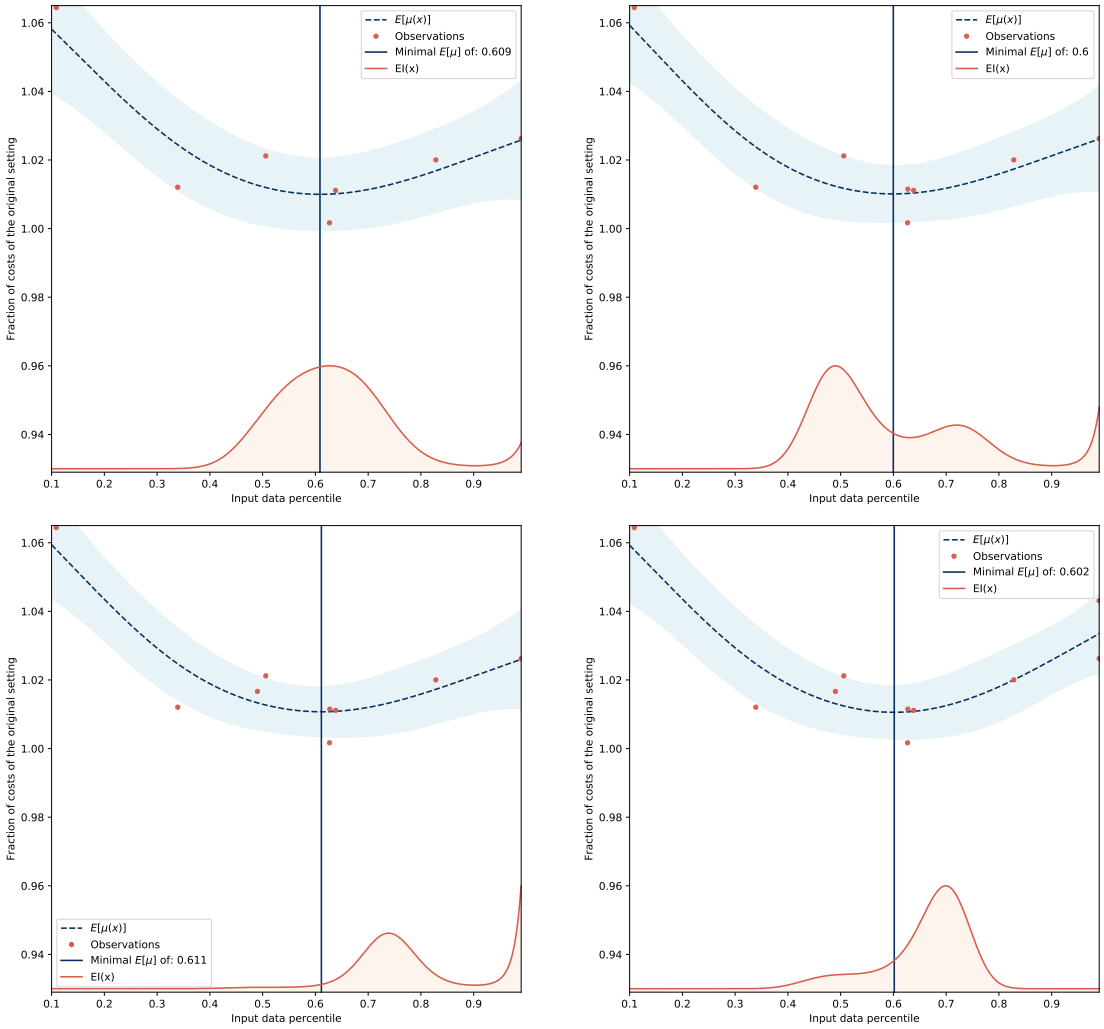

Figure 5.10: Estimated relation between handling time service level and total costs respectively after 7, 8, 9 and 10 iterations with a service cost factor of 28.57. The red line portrays the acquisition function.

The estimated cost factor of 28.57 was used in an optimization run with 4 different cases than were previously used to create a independent experiment. The optimization was run for a total of 30 runs in order to check whether 15 iterations are sufficient. To show how the acquisition function behaves during the iterations, the GP-model and corresponding Expected improvement function for iteration 7 to 10 can be seen in Figure 5.10. Here we can see that in the acquisition function chooses a point close to the current minimum after the 7th iteration, exploiting the current knowledge. After the 9th iteration a new observation is done in a section where the variance is higher, exploring the search space. This shows that the acquisition function balances exploration and exploitation automatically.

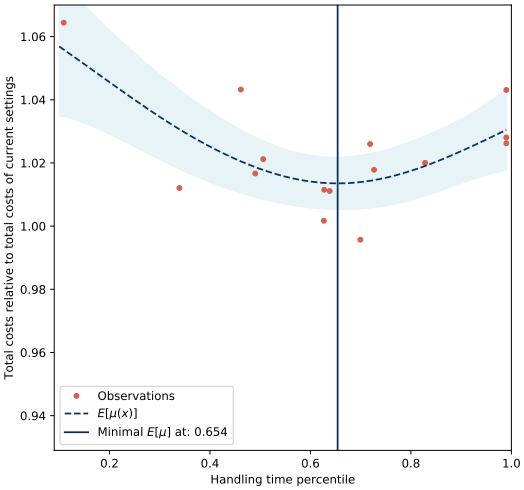

Figure 5.11: Estimated relation between handling time percentile and total costs after 15 iterations with a service cost factor of 28.57.

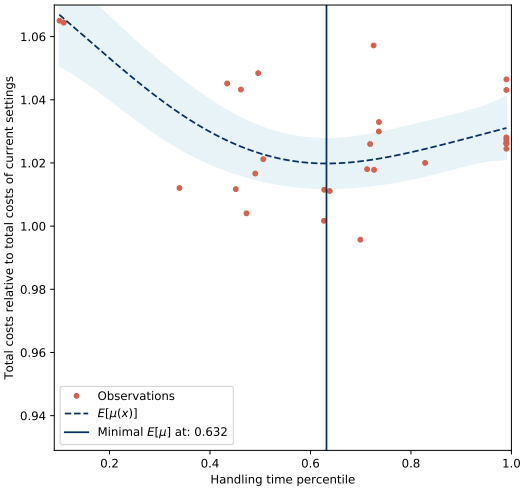

Figure 5.12: Estimated relation between handling time percentile and total costs after 30 iterations with a service cost factor of 28.57.

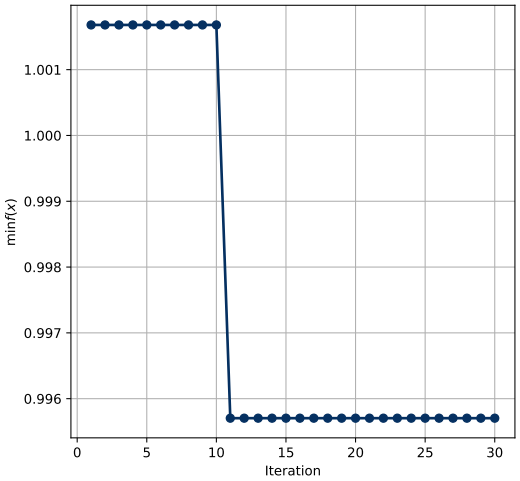

Figure 5.13: Minimal solution value of all solutions after a number of iterations.

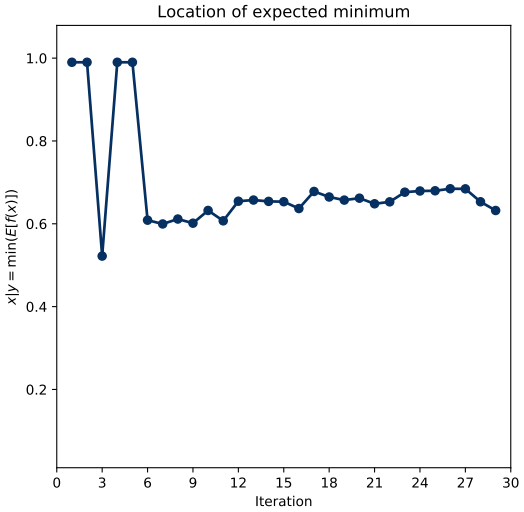

Figure 5.14: Estimated location of the minimum as approximated by the GP after a number of iterations.

In Figures 5.11 and 5.12, the results of the optimization after respectively 15 and 30 iterations of the optimization algorithm can be seen. We can see that the observation that has the lowest total costs after 30 iterations is found within the 15 iterations. We note that settings are estimated to result in higher total costs than the current handling times settings. When we look at the observations, we note that only the best observation has total costs lower than the total costs of the current handling time setting. This can also be seen in Figure 5.13, where the value of the best observations is plotted over the iterations. Only once is a new minimum value found. This is caused by the fact that the first observation is the second best measurement in the end. Since we know that our measurements are noisy, we are not only interested in the observations with the

lowest total costs, but in the location of the minimum of the estimated GP-model. This represents the setting that is estimated to have the lowest total costs in general. In Figures 5.11 and 5.12, we also see that the location of the expected minimum shifts from 0.654 to 0.632 respectively from iteration 15 to 30. This is a shift of 0.022 or 3.5%. The location of the expected minimum as estimated by the GP-model over the iterations can be seen in Figure 5.14. In the 15th to 30th iteration the location of the expected minimum varies between 0.632 and 0.685. This range has a width of 0.053. The shift of this estimated minimum is caused by two factors. The first factor that causes the minimum to shift is the fact that the GP-model is relatively flat in the part where the minimum lies. Using a handling time percentile of 0.632 or 0.685 is estimated to result in relative total costs of respectively 1.0198 and 1.0202. The second factor that causes the minimum to shift is the high variance between the observations. In Figure 5.14, we note that the location of the estimated minimum is relatively stable after the 6th iteration.

5.4 Configuration comparison

Since we had to guess the service cost factor in the final experiment in order to run the optimization algorithm, the resulting found handling time percentile might not be the best result for the Retailer. To find a good handling time percentile, we will compare the observations on more tangible KPIs.

To find the handling time percentile that gives a good general performance, we look at the relation of the plan costs and number of late pallets with the handling time percentile. This was done by using all observations from the optimization runs with a service cost factor of 10, 20, 30 and 40. The relation with the handling time setting was estimated with a Gaussian Process (GP). In Figures 5.15 and 5.16, these two relations can be seen. In Figure 5.15, we can clearly see the fluctuation of the solution quality, as there are in some cases relatively large differences in plan costs for settings that are close. To identify a region of interest, the handling time percentile at which the KPIs are approximately equal to that of the current handling time setting, are marked. For the plan costs this is 0.63, and since plan costs increase with the handling time, a handling time setting lower than 0.63 may give lower plan costs. The number of late pallets is approximately equal to the current settings at a handling time percentile of 0.58, indicating that configuring a higher handling time percentile would give a lower number of late pallets on average. These two values combined give a small range of handling time percentiles that is interesting, as both KPIs may improve over the current situation in this range.

To find a good value within that range, we evaluate the performance of handling time percentiles of 0.58, 0.6 and 0.63. To reduce the noise in the measurements caused by the planning heuristic, 10 available cases were used instead of the four that were used in the optimization runs. This is now possible as the we do not perform iterations but only run each setting once. The results of these settings and of the current configuration can be seen in Table 5.1.

We note that only the plan costs of the 0.58 handling time percentile setting, is higher than the plan costs of the current configuration. The mean number of late pallets is lower for all three proposed settings. However, the deviation in these values refrains us from drawing hard conclusions from this result. Using a one tailed t-test and testing whether the mean late number of pallets is greater for the current configuration than of the proposed 0.6 setting resulted in a

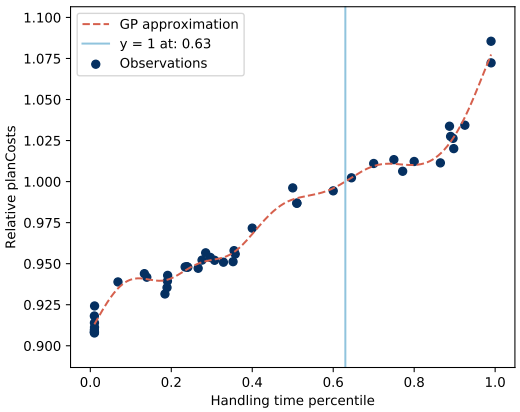

Figure 5.15: Relation between the relative plan cost and the handling time percentile estimated by a Gaussian Process.

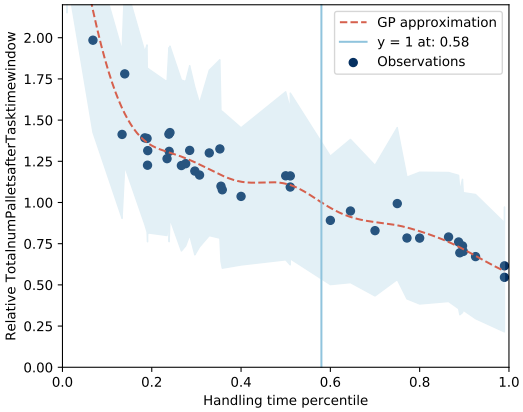

Figure 5.16: Relation between the relative number of late pallets and the handling time percentile estimated by a Gaussian Process.

Table 5.1: Results of the current setting and three handling time percentiles of interest, tested on 10 cases with 100 simulation replications. The value after ± indicates the half width of the 95% confidence interval of the mean.

Setting	**Plan costs**	**Mean number of late pallets**	**Distance (km)**	**Planned duration (hours)**	**Mean realised duration (hours)**
Current Configuration	1611786	112.5 ± 6.4	195780	11334.1	13152.2 ± 12.1
0.58	1617300	109.5 ± 6.0	197956	11325.1	13252.6 ± 12.4
0.6	1611885	107.8 ± 6.3	197494	11286.3	13085.1 ± 12.7
0.63	1609929	108.3 ± 6.2	196417	11248.5	12927.7 ± 10.3

p-value of 0.14. So, our evidence is not strong. Remarkable is that the 0.6 setting has a lower average number of late pallets than both the 0.58 and 0.63 setting. The relation in these results between the plan costs and the handling time percentile is also opposite of what is expected. This again points to fluctuating quality of solution produced by ORD. The lower costs in the proposed setting compared to the current configuration is caused by a lower total duration, as the distance of the three proposed settings is higher than the distance of the current configuration. Of the three proposed settings we favour the input setting with a handling time percentile of 0.63 as it has a reduced plan cost and a reduced number of time window violations. The 0.63 handling time percentile reduces the plan costs with 0.12% and the number of missed pallets with 3.73%.

In Table 5.2, the current configuration can be seen compared to the proposed setting. We can see that the proposed new setting increases the fixed duration and decreases the variable duration for the first three address types, while the reverse is true for the last two address types.

As we have stated in Section 1.4, no new settings are added, so only the existing settings will be tuned. However, creating a reparametrization model for the individual address level is easily done and allows us some insight into the variation that is present among addresses belonging to the same address types. Figure 5.17 gives a graphical representation of the settings that a reparametrization model for the handling time settings of individual addresses estimates,

Table 5.2: Current configuration for all address kinds compared to the new proposed setting. All durations are given in minutes.

Address kind	**Parameter type**	**Current setting**	**Proposed setting**
SHOPRNDYD	Fixed time	30	36.3
	Time per pallet	3	2.2
shop	Fixed time	25	31.3
	Timer per pallet	3	2.7
shop RND	Fixed time	30	30.7
	Time per pallet	3	2.9
RND_45	Fixed time	45	26.5
	Time per pallet	3	4.2
RND_60	Fixed time	60	24.0
	Time per pallet	3	4.1

compared to the settings of the address kind. We first of all note, that the settings of the individual addresses are spread over the graphs. This indicates that there might still be some performance improvements to be gained if additional settings are added instead of only improving current ones. We also note that the setting on the address type level is not simply an average of the settings of the address levels. For the address types with a lot of addresses, the settings on the address level have relatively high variable time per pallet compared to the address kind setting. This may be caused by the fact that there is less data available for the individual addresses, which causes these to be more susceptible to duration changes when the number of pallets change. On the address kind level these are smoothed out by the multiple addresses. This should be considered when attempting to find settings for individual addresses.

5.5 Conclusion

This chapter provides method and results of the optimization of the inputs of the simulation.

We have shown how the data can be used in the optimization process. This was done by creating a model that gives an estimated distribution of the input parameters. Instead of optimizing over every separate value, we use the percentile as input and use the reparametrization model to find the accompanying input parameters. This allows for much easier optimization as the dimensions are reduced. Besides the easier optimization, the reparametrization model itself can be used to quickly gain insight into what the parameters look like according to the data.

We have shown the set up of a sequential optimization algorithm. A Gaussian Process was implemented to estimate the relation between the input and output of the simulation. Using the Expected Improvement as acquisition function automatically balances exploration and exploitation.

Using a logical weight to combine the two KPIs into a single total cost KPI did not result in a balanced optimization result. A weight was estimated by finding the weight that minimized the distance between the current situation and the best found setting. Using this weight, we have shown that the optimization algorithm estimates the location of the minimum with a deviation

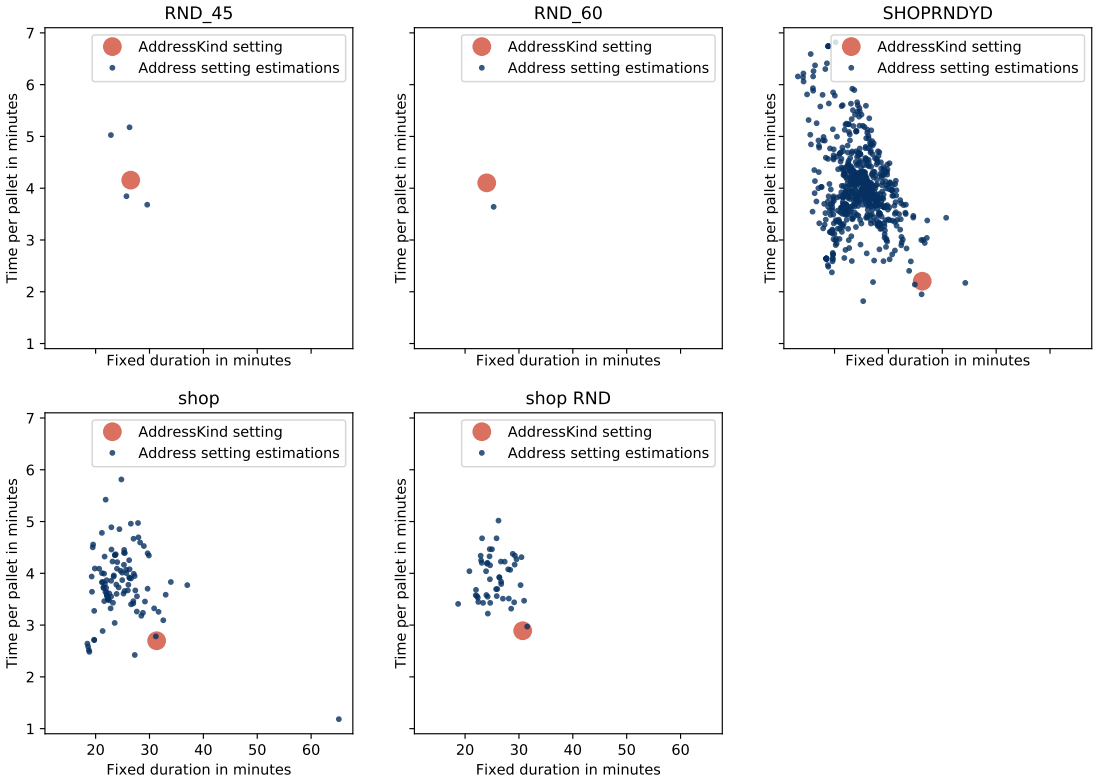

Figure 5.17: The estimated settings per address compared to the setting estimated for the address kind for a handling time percentile of 0.63.

of 3.5% of the minimum that is estimated after 30 iterations. The observation with the lowest total costs is found within 15 iterations. We have seen that there is noise in the KPIs of both the planning and the simulation.

An alternative method was demonstrated to identify an interesting region and find a good setting based on tangible KPIs. The resulting settings that have been investigated show that the number of time window violations can be decreased while plan costs remain approximately equal. From these investigated settings, the handling time percentile of 0.63 is recommended as setting to implement. This setting reduces the plan costs with 0.12% and the number of missed pallets with 3.73%.

Finally, we have shown that there are still opportunities for improvement by looking into setting handling time settings per address.

Chapter 6

Implementation

In this chapter, we will explain how the method that was applied in this thesis on the Retailer case, can be implemented by ORTEC so that it can be applied to other customer cases. This chapter is divided so that each section describes the implementation of a part of the methodology. The method is divided into parts as shown in Figure 4.1. In Section 6.1, the required parts needed for implementing the data preprocessing step are listed. Section 6.2 gives some suggestions on the implementation of the distribution fitting step. In Section 6.3, the required steps for implementing a simulation model is given. Section 6.4 describes how the optimizing of the simulation can be implemented. Section 6.5 concludes the chapter.

In the description of the implementation we make a distinction between steps in the method that have to be tailor made for every customer and the steps that are general and can be implemented in the software.

6.1 Data preprocessing

The first part of the method is the preprocessing of the data. In this part the available data on the planning and realisation has to be joined, extracted and cleaned. Since the data at the client can differ in source and granularity, this step needs attention of a consultant that applies the methodology. It can be beneficial, if a standardized template for the query is made for common forms of implementation and realisation data. This can greatly reduce the total duration of the project. In this stage the realisation data has to be joined with the planning data. This join has to occur on the least detailed level of these two parts. In our case, the realisation data dictated that the data set had a full stop or travel action in each row of the set. When more detailed realisation data is available, for instance, durations per road section or separate waiting and unloading times for the stops, this has to be taken into account in the data extraction process. During this step, communicating with the client is an important aspect to gather information on suspicions that they have.

After a preliminary data set has been created, features that might be of interest have to be created. This is done using a priori knowledge on what factors could influence the durations of interest. This a priori knowledge comes from suspicions voiced by the customer, the way the durations are configured in the software and some common sense. This process is shown

in Figure 4.1 as a single step which is performed once, but this creation of features occurs in parallel of the entire method as during the process the knowledge on the system grows and new insights arise.

When the features have been created, the data should be inspected and cleaned. This can be done using standard data science methods. For the cleaning of the data, we recommend the use of the 2-sided MAD method, as it is robust and able to deal with skewed distributions. To decide on which parameters to tune, the cleaned data set should be briefly investigated to see which durations deviate a lot. The granularity of the data combined with the availability and quality should also be considered when deciding which parameters to tune. For instance, in our case we chose not to investigate the travel times or speeds, as only total durations of the trips were available and the configuration required speeds per road type.

6.2 Distribution fitting

After the data is extracted and cleaned, distributions can be fitted to the data. First should be decided what distributions are needed, in our case this was for the travel and stop durations and the start time deviations. However, other types of durations, like cleaning or (un)coupling durations, might also be required. When the query has been standardized, a script for this process can be written once and used for multiple customers. We recommend making this script modular so that new distribution models can easily be added.

For each set of distributions, the exploration and fitting process is roughly the same. We consider the methods employed in this thesis to be quite general. One does not necessarily have to make the distinction between statistical and advanced statistical methods or choose the same method that we have chosen. However, the process of first investigating explanatory features and then fitting and testing the distributions models, is a general data science process that can be applied anywhere. We do think it is useful to fit multiple models to see which has the better overall performance. Also, the usage of a RF to calculate feature importance and use this in the exploration of the features is beneficial in our opinion. In our opinion, the RFCDE is a useful model to investigate as it is versatile and easy to tune and train. Also, the ability to deal with categorical variables makes it a useful technique.

6.3 Simulation model

Programming the simulation model is combining the distributions with the assumptions. These assumptions can either come from process experts or from limitations that are set by the data availability. The most important part is to carefully list all these assumptions so that one later can understand why a model might not give the desired results. These methods are again quite generally applicable to other cases.

Before anything can be simulated, case files need to be created. In this thesis, this was done by creating a script that used a separate environment to add new orders with the same characteristics as the historic orders. This can more easily be done when an option is implemented in the scheduling module that allows completed orders to be extracted from the plan so that

these can be used in the optimizer. Implementing this is not only beneficial for tuning input parameters as we have done, but can also help create realistic case files for tuning the algorithm.

A first step in the implementation of the simulation is parsing the planning files that ORD creates, to files that can be used as input for the simulation. Using the right data structure as input for the simulation, allows for a more readable script that performs the simulation. In our case, the output of ORD contained the stop actions per order and the travel times were added to the first order in a stop. This was converted to a set of routes that contained separate travel actions and joined the actions occurring at the same location. This allowed the simulation to simply loop over this list and use the attributes of the action at hand to simulate a duration. As it is useful to standardize the query, it would also be beneficial to create a standard output format that can be used for the simulation.

Standardizing the simulation script itself is only beneficial when the output format of ORD is also standardized. The script that was written to run the simulation in this thesis forms a good starting point. It can be further improved in terms of computational speed by running the simulation in multiple processes for every case. A requirement for this is that the distribution models can all be loaded from the disk. This was not the case for the RFCDE and therefore this was not yet implemented.

When the model has been created, a validation has to be performed. In this thesis, this was done by simulating the routes as they were planned in the data set and comparing the simulated realisations to the historic simulations. This can also be done in other cases.

6.4 Simulation optimization

The first step in the entire simulation optimization process was creating a model that reparametrizes the input data values to a single value. This is possible in other cases if some conditions are met. First, the data should be at least as detailed as the parameters that are investigated. Secondly, the model used for determining the duration during planning is a linear combination of features that are also available in the data set. When these two conditions are satisfied, it is possible to make a reparametrizing model. When during the distribution fitting it is discovered that the durations depend on a lot more features than are used in planning, some strategies can be used to overcome this. In our case the stop durations depend on the estimated waiting time as this was included in the realisation data but not in the planning entity, setting this feature to 0 in the reparametrization effectively filters out the effect that this feature has on the duration. Other features that cannot be set to 0 can simply be left out the model as this causes the model to use the average effect that the left out feature has. If multiple sets of durations are investigated, like stop and travel durations, one should consider creating separate reparametrization models to be able to tune the service levels separately. Creating such a reparametrization model is relatively fast compared to running the simulation. When the methodology is applied at a customer to tune the input parameters periodically, one might chose to use the percentile that was found previously and only fit a reparametrization model on the latest data to find new parameters. This process could be automated to run without supervision on moments when the computational load is low. This would greatly reduce the computational resources needed to find new parameters.

The method used to optimize the simulation is generally applicable and could be used in other cases. Although it was not demonstrated in this thesis, BO is able to deal with categorical and multi-dimensional optimization problems. Also, the alternative method that was demonstrated to find an interesting region of handling times is applicable to other cases. For this any combination of KPIs can be used to determine what range of values is interesting.

6.5 Conclusion

We have shown in this chapter that the methodology that developed and applied in this thesis is generally applicable to other customer cases. Also, some insights into how the methodology that was developed in this thesis can be implemented have been given. The steps of the methodology regarding the data exploration, require the attention of a consultant occupied with tuning the input parameters. However, standardization in the right places can greatly reduce the time needed to complete an input parameter tuning project. This standardization can be achieved by productizing this methodology.

Chapter 7

Conclusions and recommendations

The answers to the five research questions stated in Section 1.3, collectively fulfil our research goal. The answers to these questions have been given in Chapters 2 to 6. For a recap on the answers to the research questions we refer to the (sub)sections containing conclusions in these chapters. In Chapter 2, the context in which the research is performed and the routing case that is present at the Retailer have been described. Chapter 3 gives a review on the available literature on the routing problem and the methods that have been used to tune the input data. Chapter 4 describes the process of creating a simulation model from the data set. This simulation model is used to find new input data values in Chapter 5. Finally, in Chapter 6 we have given a description of the implementation of the methodology so that it can be applied at other customers.

In this chapter we conclude this research. We report our findings in Section 7.1 and discuss these in Section 7.2. Finally, in Section 7.3, we lay out our recommendations based on the obtained results.

7.1 Conclusion

The goal of this research was to improve planning by using realisation data. This was formulated in our research goal as:

> *Develop an offline learning method that finds better input settings to improve plannings made by ORD*

We have developed a generally applicable method that can be used to systematically approach the problem and can find better input settings, even when the simulation model has to make some assumptions that are violated in the real system.

With this method, we have shown that the use of data analytics can improve the performance of vehicle routing algorithms. We have shown how data from the real system can be used to adjust the problem definition so that the overall resulting solution improves. From this, we can conclude that for optimization projects, besides the improvement of the problem solving method, also the improvement of the problem definition is important. As heuristic methods that are in use

are already quite complex, the use of data analytics can yield an increase in routing performance relatively easily. This makes a small step towards the integration of data science and OR.

The methodology uses the realisation data to create a cleaned data set on which several distribution models are fitted. We have seen that the cleaning of the data set is important, as there can be many samples having large deviations caused by unknown irregular circumstances. The Random Forests for Conditional Density Estimation (RFCDE) was demonstrated as a practical, well performing distribution model. The ability of the RFCDE to deal with categorical variables of high cardinality and easy tuning of hyper-parameters makes it suitable for the task.

We validated the simulation by comparing historic durations to simulated durations when the same planned routes are used. We have seen that the quality and availability of data dictate the accuracy of the simulation model, as was seen in our customer case. However, even when some assumptions have to be made that cause the simulation model to give unrealistic results, the method is still able to use it.

We have shown that a sequential Bayesian Optimization algorithm can be used to find good handling time settings. This was achieved by using a reparametrization model for the input data to reduce the dimensionality of the optimization problem. In the optimization of the simulation model, fluctuating solution quality of the plannings can give relatively large differences in KPIs for settings that are relatively close. This confirms that when we want to solve complex VRP instances quickly, no guarantees can be given in the solution quality. This also indicates that over fitting could occur if a limited number of cases is used. The optimization algorithm was able to find the location of the estimated minimum in 15 iterations with an error of 3.5% compared to the location of the estimated minimum after 30 iterations.

A method for determining input setting regions of interest was shown, for when the weights with regard to the plan costs and the time window violations are unknown, as was the case in our research. This method is able to identify a region where all KPIs are expected to improve. However, the solution with the lowest total costs may be outside this region. This is dependent on how KPIs are valued by the client.

Finally, a proposed input setting was shown to yield 0.12% lower plan costs and 3.73% less time window violations as compared to the current settings used by the Retailer. This might be improved further by looking into adding more detailed settings.

7.2 Discussion

All findings with respect to performance of the proposed setting are dependent on the data set that was used. This means that the strength of these finding stands or falls by the quality of the data set. First of all, there might still be correlation between durations of stops and travels within a route that was not found during the fitting of the distributions. This could be caused by features that were missing, either as these were not created or because data is simply not present in the database.

We have seen that the solution quality of the plannings made by ORD, can cause longer handling time settings to yield plannings having shorter distance and total duration than plannings made with shorter handling time settings. This raises the question of what the real underlying performance is of the setting and when we can be sure of measuring it correctly, as a setting that

gives a good result on one case may be bad on another. This problem could be circumvented by tuning the planning algorithm to give a more stable performance or using more cases to get a better average. However, in order to get a more stable performance of the planning algorithm, the running time will increase. So, this trade-ff between the accuracy of the measurement and the computation time of a single measurement is worth exploring in order to improve the performance of the simulation optimization algorithm. Also, more insight on what cases cause some of the settings to have a bad performance could prove useful.

Data availability can force assumptions on the simulation model that are violated in the real system. We had to make the assumption that all routes were independent, as no reliable data was present on the loading at the depot. This assumption of independence is almost certainly not valid and can cause our simulation model to be optimistic about the number of time window violations that are made. However, since all comparisons are made within the same simulation and given the fact that our proposed solution improves both KPIs, we can say that the proposed input setting is an improvement compared to the current configuration.

7.3 Recommendations

We first of all recommend that the Retailer investigates the proposed new handling time setting to see whether performance improves. Secondly, to be able to create a more valid simulation model and tune the durations on the depot, the data collection of the durations at the depot should be improved.

In order to make this type of research easier to perform, we recommend that ORTEC looks into creating standard queries to extract and join realisation and planning data. When this is achieved, quick insights into the ranges of parameters can be created by training a reparametrization model.

To determine the best percentile for input data values at a customer, simulation can be used. To be able to find the best setting in that simulation, we recommend discussing the KPI or restriction on the service level that accurately represents the goal, with the customer.

When a service level is determined for a customer, the reparametrization model could be used periodically to update the input parameters, as it is relatively cheap to compute compared to the simulation model.

To get a better understanding of why some deliveries are late, explicit information on missed deliveries could be logged in the data base. This might enable data analysts at the customer to notice structural deviations quicker.

Another interesting direction for further research is to look into configuring more detailed input parameters. As can be seen in Figure 5.17, the more detailed settings can vary quite a lot, indicating that there is still room for improvement.

Bibliography

Amaran, S., Sahinidis, N. V., Sharda, B., and Bury, S. J. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, 2016.

Ambrogioni, L., Güçlü, U., van Gerven, M. A. J., and Maris, E. The Kernel Mixture Network: A Nonparametric Method for Conditional Density Estimation of Continuous Random Variables. 2017.

Archetti, C. and Speranza, M. G. Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19(1-2):3–22, 2012.

Baldacci, R., Battarra, M., and Vigo, D. Routing a Heterogeneous Fleet of Vehicles. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 3–27. Springer US, Boston, MA, 2008.

Banks, J., Carson, J. S., Nelson, B. L., and Nicol, D. *Discrete-Event System Simulation*. Prentice Hall, 5 edition, 2010.

Bhatnagar, S., Prasad, H., and Prashanth, L. *Stochastic Recursive Algorithms for Optimization*, volume 434 of *Lecture Notes in Control and Information Sciences*. Springer London, London, 2013.

Bishop, C. M. Mixture Density Networks. Technical report, Aston University, 1994.

Bishop, C. M. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116, 1995.

Botev, Z. I., Grotowski, J. F., and Kroese, D. P. Kernel density estimation via diffusion. *Annals of Statistics*, 38(5):2916–2957, 2010.

Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and regression trees*. Chapman & Hall, 1984.

Brochu, E., Cora, V. M., and de Freitas, N. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. 2010.

Chang, H. S., Fu, M. C., Hu, J., and Marcus, S. I. Simulation-based Algorithms for Markov Decision Processes. In *Communications and Control Engineering*, number 9781846286896 in Communications and Control Engineering, pages 1–189. Springer London, London, 2007.

Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., and Semet, F. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5):512–522, 2002.

Cousins, C. and Riondato, M. CaDET: interpretable parametric conditional density estimation with decision trees and forests. *Machine Learning*, 2019.

Cox, D. R. Tests of Separate Families of Hypotheses. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 105–123, Berkeley, Calif., 1961. University of California Press.

Dantzig, G. B. and Ramser, J. H. The Truck Dispatching Problem. *Management Science*, 6(1): 80–91, 1959.

Demkes, K. H.-j. Automated tuning of an algorithm for the vehicle routing problem. *Masters Thesis*, 2014.

Dror, M. and Trudeau, P. Savings by Split Delivery Routing. *Transportation Science*, 23(2): 141–145, 1989.

Elder IV, J. F. and Pregibon, D. A statistical perspective on knowledge discovery in databases. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, chapter A Statisti, pages 83–113. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

Frazier, P. I. A Tutorial on Bayesian Optimization. 2018.

Frazier, P. I., Powell, W. B., and Dayanik, S. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

García, S., Luengo, J., and Herrera, F. Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowledge-Based Systems*, 98:1–29, 2016.

Golden, B., Assad, A., Levy, L., and Gheysens, F. The fleet size and mix vehicle routing problem. *Computers and Operations Research*, 11(1):49–66, 1984.

Gromski, P., Xu, Y., Kotze, H., Correa, E., Ellis, D., Armitage, E., Turner, M., and Goodacre, R. Influence of Missing Values Substitutes on Multivariate Analysis of Metabolomics Data. *Metabolites*, 4(2):433–452, 2014.

Gudgeon, A. C. and Howell, D. C. Statistical Methods for Psychology. *The Statistician*, 43(1): 211, 1994.

Hall, M. A. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, 1999.

Hampel, F. R. The breakdown points of the mean combined with some rejection rules. *Technometrics*, 27(2):95–107, 1985.

Hawkins, D. M. The Problem of Overfitting, 2004.

Hill, S. D. and Fu, M. C. Transfer optimization via simultaneous perturbation stochastic approximation. In *Winter Simulation Conference Proceedings*, pages 242–249. IEEE, 1995.

Ho, T. K. Random Decision Forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278—-, Washington, DC, USA, 1995. IEEE Computer Society.

Hsiao-Mei, W. Comparison of the Goodness-of-Fit Tests: the Pearson Chi-square and Kolmogorov-Smirnov Tests. *Ling Tung University*, 6(1):57–64, 2009.

Hu, C., Lu, J., Liu, X., and Zhang, G. Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers and Operations Research*, 94:139–153, 2018.

Izbicki, R. and Lee, A. B. Converting high-dimensional regression to high-dimensional conditional density estimation. *Electronic Journal of Statistics*, 11(2):2800–2831, 2017.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Kaczynski, W., Leemis, L., Loehr, N., and McQueston, J. Nonparametric random variate generation using a piecewise-linear cumulative distribution function. *Communications in Statistics: Simulation and Computation*, 41(4):449–468, 2012.

Kim, S. Gradient-based simulation optimization. In *Proceedings - Winter Simulation Conference*, pages 159–167, 2006.

Kleinberg, E. M. An overtraining-resistant stochastic modeling method for pattern recognition. *Annals of Statistics*, 24(6):2319–2349, 1996.

Laporte, G., Louveaux, F., and Mercure, H. Vehicle routing problem with stochastic travel times. *Transportation Science*, 26(3):161–170, 1992.

Law, A. M. A tutorial on how to select simulation input probability distributions. In *Proceedings - Winter Simulation Conference*, pages 103–117. IEEE, 2011.

Law, A. M. and Kelton, W. D. *Simulation modeling and analysis*, volume 2. McGraw-Hill International Education, 5th edition, 2015.

Leys, C., Ley, C., Klein, O., Bernard, P., and Licata, L. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

Li, X., Tian, P., and Leung, S. C. Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production Economics*, 125(1):137–145, 2010.

Liu, H., Shah, S., and Jiang, W. On-line outlier detection and data cleaning. *Computers and Chemical Engineering*, 28(9):1635–1647, 2004.

Maimon, O. and Rokach, L. Introduction to Knowledge Discovery and Data Mining. In *Data Mining and Knowledge Discovery Handbook*, pages 1–15. Springer US, Boston, MA, 2010.

Meinshausen, N. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

Mes, M. R. K., Powel, W. B., and Frazier, P. I. Hierarchical Knowledge Gradient for Sequential Sampling. *Journal of Machine Learning Research*, 12:2931–2974, 2011.

Montgomery, D. C. and Runger, G. C. *Applied Statistics and Probability for Engineers Problem solutions*. Wiley, 2003.

Norwood, B., Ferrier, P., and Lusk, J. Model selection criteria using likelihood functions and out-of-sample performance. In *NCR-134 Conference on Applied Commodity Price Analysis, Forecasting, and Market Risk Management*, page 19 p., 2001.

Ordóñez, F. Robust Vehicle Routing. In *Risk and Optimization in an Uncertain World*, pages 153–178. INFORMS, 2010.

Ostermeier, M., Martins, S., Amorim, P., and Hübner, A. Loading constraints for a multi-compartment vehicle routing problem. *OR Spectrum*, 40(4):997–1027, 2018.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100, 2017.

Pospisil, T. and Lee, A. B. RFCDE: Random Forests for Conditional Density Estimation. *ArXiv*, abs/1804.0, 2018.

Powell, W. B. The Knowledge Gradient for Optimal Learning. In *Wiley Encyclopedia of Operations Research and Management Science*. American Cancer Society, 2010.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

Rieck, J. and Zimmermann, J. A new mixed integer linear model for a rich vehicle routing problem with docking constraints. *Annals of Operations Research*, 181(1):337–358, 2010.

Rothfuss, J., Ferreira, F., Walther, S., and Ulrich, M. Conditional Density Estimation with Neural Networks: Best Practices and Benchmarks. 2019.

Ryzhov, I. O., Powell, W. B., and Frazier, P. I. The knowledge gradient algorithm for a general class of online learning problems. *Operations Research*, 60(1):180–195, 2012.

Sargent, R. G. Verification and Validation of Simulation Models. In *Proceedings of the Winter Simulation Conference*, WSC '11, pages 183–198. Winter Simulation Conference, 2011.

Scott, D. W. and Sain, S. R. Multi-dimensional Density Estimation. 2004.

Silverman, B. W. *Density estimation: For statistics and data analysis*. Number 1951. Monographs on Statistics and Applied Probability, 1986.

Smyth, P. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and Computing*, 10(1):63–72, 2000.

Spall, J. C. Stochastic Approximation and the Finite-Difference Method. In *Introduction to Stochastic Search and Optimization*, pages 150–175. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2005.

Stekhoven, D. J. and Bühlmann, P. Missforest-Non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Toth, P. and Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, 2014.

van der Herten, J., Couckuyt, I., Deschrijver, D., and Dhaene, T. Fast Calculation of the Knowledge Gradient for Optimization of Deterministic Engineering Simulations. *arXiv:1608.04550*, pages 1–17, 2016.

Wang, X. and Regan, A. C. Assignment Models for Local Truckload Trucking Problems with Stochastic Service Times and Time Window Constraints. *Transportation Research Record: Journal of the Transportation Research Board*, 1771(1):61–68, 2007.

Zaman, Q., Azam, M., Khusro, S., Iqbal, M., and Pfeiffer, K. P. Fundamentals of Classification and Regression Trees. *Elixir Bio Tech*, 40:5407–5414, 2011.

Zunic, E., Hindija, H., Besirevic, A., Hodzic, K., and Delalic, S. Improving Performance of Vehicle Routing Algorithms using GPS Data. In *14th Symposium on Neural Networks and Applications, NEUREL 2018*, pages 1–4. IEEE, 2018.

Appendix A

Data description

Table A.1: Variables present in data set only present for travel actions

Variable name	**type**	**description**	**calculation**
startAddressId	integer	identifying number for the starting address	
startAddressKindName	string	type of the starting address	
isStartDepot	boolean	indicating whether travel starts at the depot	
finishAddressId	integer	identifying number for the destination address	
finishAddressKindName	string	type of the destination address	
isFinishDepot	boolean	indicating whether travel ends at the depot	
stopActionStopNumber	integer	number indicating the ordinal rank of the travel in the trip	
MaxStop	integer	number indicating total number of stops in the trip	
NumStopsLeft	integer	number of stops remaining in the trip	MaxStop - stopActionStopNumber
distance	float	total distance of travel in kilometers	

Table A.2: Variables present in data set only present for stop actions

Variable name	**type**	**description**	**calculation**
finishAddressId	integer	identifying number for the address the stop takes place	
finishAddressKind	integer	identifying number for the type of address the stop takes place	
isFinishDepot	boolean	indicating whether stop takes place at the depot	
stopActionStopNumber	integer	number indicating the ordinal rank of the stop within the trip	
MaxStop	integer	number indicating total number of stops in the trip	
NumStopsLeft	integer	number of stops remaining in the trip	MaxStop - stopActionStopNumber
tot_num_pal_deliver	float	Total number of pallets delivered in stop	
tot_num_pal_pickup	float	Total number of pallets picked up in stop	
plannedWaitStart	datetime	planned arrival at stop location	
plannedWaitFinish	datetime	planned start of actual unloading	
plannedWaitTimeMinutes	float	Duration of planned waiting time	plannedWaitFinish - plannedWaitStart, default=0
plannedDuration WithoutWaitingTime	float	Duration of action without waiting time	plannedDurationMinutes - plannedWaittimeMinutes
MaxStartDate	datetime	start of time window	
MinTillDate	datetime	end of time window	
MinutesPlannedStart BeforeStartTimeWindow	float	number of minutes the action is planned to start before the start of time window, 0 if planned start is within time window	max(0, maxStartDate - plannedStartInstant)
MinutesPlannedFinish AfterEndTimeWindow	float	number of minutes the action is planned to end after the end of time window	max(0, plannedFinishinstant - MinTillDate)
MinutesRealisedStart BeforeStartTimeWindow	float	number of minutes the action actually started before the start of time window	max(0, maxStartDate - realisedStartInstant)
MinutesRealisedFinish AfterEndTimeWindow	float	number of minutes the action actually ended after the end of time window	max(0, realisedFinishinstant - MinTillDate)

Table A.3: Variables present in data set for both stop and travel actions

Variable name	type	description	calculation
id_action	integer	unique identifier number for a single action	
id_shift	integer	unique identifier number for a trip	
id_resource1	integer	Truck id number	
resourceKindCode1	string	Truck type	
plannedStartInstant	datetime	date and time of planned start of action including waiting times if present	
plannedFinishInstant	datetime	date and time of planned finish of action	
dayFromStart	integer	Number of days since earliest action present in database	
realisedStartInstant	datetime	date and time of actual start of action	
realisedFinishInstant	datetime	date and time of actual finish of action	
plannedDurationMinutes	float	planned duration of action in minutes	plannedFinishInstant - plannedStartInstant
realisedDurationMinutes	float	realised duration of action in minutes	realisedFinishInstant - realisedStartInstant
DurationDeltaMinutes	float	delta of duration in minutes, positive means action took longer than planned	realisedDurationMinutes - plannedDurationMinutes
RelativeDurationDelta	float	relative duration delta	DurationDeltaMinutes / plannedDurationMinutes
StartTimeDeviationFromPlanMinutes	float	delta of the start time in minutes, positive means action started later than planned	realisedStartInstant - plannedStartInstant
weekdayRealisedStart	integer	integer of weekday of realised start of action (Monday=1,..Sunday=7)	
weekdayRealisedFinish	integer	integer of weekday of realised finish of action (Monday=1,..Sunday=7)	
hourRealisedStart	integer	hour of realised action start	
hourRealisedFinish	integer	hour of realised action finish	