Optimizing smart waste collection through container selection

Jelmer van Zeijl

Delft University of Technology

Master:	Mechanical Engineering
Track:	Transport Engineering and Logistics
Assignment:	Graduation Project
Course Code:	ME54035

Optimizing smart waste collection through container selection

Author:
Jelmer van Zeijl

Thesis committee:
Prof. dr. ir. R.R. Negenborn, committee chair
Dr. Bilge Atasoy, University supervisor
Ir. J.Y. Brandt, Company supervisor

Date: February 2, 2021
Student number: 4740742
Rapport number: 2020.MME.8482

Preface

With this thesis I will complete my Master of Science in Mechanical Engineering at the University of Technology in Delft. Withing the master of Mechanical Engineering I have specialized in the track of Transport Engineering and Logistics. In this thesis I will cover the topic of the smart waste collection problem and the optimization of it using different container selection methods. Although this topic was exactly what I wanted to do for my graduation project, it was challenging nevertheless. I am happy with the obtained results and everything I have learned during the entire process.

I would like to start with thanking staff of AMCS for all the help and providing such an welcoming and open environment in which I could work on my thesis. This is true for the office in Rotterdam, staff located elsewhere in the Netherlands and even abroad. It was amazing to see that so many people were interested in the project and were open for questions and help whenever needed. In special I would like to thank Jelmer Brandt for guiding the project within AMCS and helping me with all data related issues. And special thanks to Friso van Wassenaer, who as a fellow graduate student at AMCS was always open for questions and discussions.

Furthermore, I would like to thank my graduation committee members from the TU Delft. First of all many thanks to Bilge Atasoy, for helping me not only with the programming for and writing of my thesis but also for being positive during meetings which gave me new motivation every time again. And lastly, Rudy Negenborn, for all his feedback during meetings which helped me step by step towards the final result.

Finally, I could not have completed this thesis without the help and support of my parents, roommates and friends. Who all had their contribution some for in depth help and others for overall support.

Jelmer van Zeijl Delft, February 2, 2021

Abstract

A significant growth of the world's population together with fast growing urbanization is causing challenges for cities in the future. One of these challenges is how to deal with waste production and therefore the waste collection in such areas. Smart waste collection is a promising solution since it optimizes an already excising infra structure including vehicles and since the collection and transportation of the produced waste accounts for roughly 70% of the waste management costs. Smart waste collection is a routing problem and can be categorized both as a vehicle routing problem (VRP) or as an inventory routing problem (IRP) depending on which container selection method is used. Smart wast collection uses sensors to measure and communicate the fill levels of the waste containers. This data can be used to optimize the process of waste collection and optimize the chosen KPIs. When all fill levels are known, routes can be optimized and containers can be collected at exactly the right time. However, literature shows a shortcoming in data treatment and the use of real data. This article will use real data obtained from the containers collected by OMRIN with the help of AMCS and their software. Another gab in literature has to do with the collection methods. Containers can be selected for collection, roughly based on three methods. The first works by setting a threshold level on the fill level of the containers. When a fill level exceeds the threshold level is will be selected for collection. The second method is called attractiveness. Attractiveness bases the collection of containers on how attractive they are. The attractiveness of a container can however be specified in many different ways. The third and last method is called must-go may-go, as named in literature. This method combines the previous ones and containers that pass the threshold level have to be collected and therefore are called must-go containers. Other containers can, based on their attractiveness, be added to a route when it is cheaper or quicker to collect that container today instead of tomorrow. The interesting thing is that these three methods have not yet been compared to each other. Therefore the main goal of this research is to compare the three aforementioned methods based on real data and investigate certain tuning parameters to optimize each model based on the chosen KPIs. The models discussed in this research are build using basic concepts used in VRPs an IRPs as well as many details, enable to represent a real SWCP as close as possible. Many details used by AMCS are also applied to the models of this research. The models are first individually optimized by changing their tuning parameters and keeping the overflow in an acceptable range of the baseline, calculated based on real data. Overall a strong negative correlation is found between the total traveled time and the amount of overflows. Furthermore a warm-up period of three days is used, meaning the first three days of a test instance will be removed in order to capture only the steady state of the models. The individual optimization shows the best solutions for a 1% threshold buffer for the threshold model, a three-day horizon with 110% upper limit for the attractiveness model and a three-day horizon with an threshold buffer of 7% and a 110% upper limit for the must-go may-go model. When compared the attractiveness model shows to have the smallest total travel time. However, its computational time exceeds the total travel time. Meaning it takes longer to calculate the collection of containers than to actually collect them in real life. The must-go may-go model shows a slightly higher total travel time, but has a significant lower computational time. Therefore making it more suitable for real world application. This research as well shows that the two-day horizon instances of both the attractiveness model and the must-go may-go model barely improve the solution of the one-day horizon. Finally a true forecasting model was used to see what potential lies with designing a detailed forecasting model, which shows to be in the same order of improvement as was obtained by tuning parameters of each model.

List of abbreviations

ALNS	Adaptive Large Neighbourhood Search
AMCS	Advanced Manufacturing Control Systems
BSA	Backtracking Search Algorithm
CPP	Chinese Postman Problem
CVRP	Capacitated Vehicle Routing Problem
CVRPTW	Capacitated Vehicle Routing Problem with Time Windows
GIS	Geographic Information System
ILP	Integer Linear Programming
IRP	Inventory Routing Problem
KPI	Key Performance Indicator
MILP	Mixed Integer Linear Programming
NP	Nondeterministic Polynomial
PSO	Particle Swarm Optimization
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
WCP	Waste Collection Problem

Contents

1 Introduction

1.1 Smart waste collection

At the moment, more than half of the world's population lives in urban areas. By 2050, the United Nations expects this amount to have increased to two-thirds of the world's population (Nations, 2014; of-Economic-and Social-Affairs, 2017). Add to this the forecast of population growth from 7.7 billion people now to 9.7 billion in 2050 and a major challenge arises for the cities of the future. The concept of smart cities arises to solve the problems expected in the future. Smart cities use the Internet Of Things (IoT), an extension of the internet in which devices, sensors, and machines form a network in which the exchange of data can take place, to be able to manage and efficiently use assets, resources and services. One of these services or challenges is how to deal with the increasing amount of waste production. By realizing that most of our waste is composed out of materials that are not inexhaustible and could be recycled and/or re-used, waste no longer is a leftover which we dump and need to get rid of. The European Union (EU) has, as part of its Circular Economy, set targets for recycling 65% of municipal and 75% of packaging waste by 2030 (EuopeanCommissionEnviroment, 2019). By increasing awareness that our waste can have environmental and economical benefits that could be taken advantage of, the interest in waste collection has grown as well. Generally, municipalities are responsible for the complete waste management system, which consists of collecting, transporting, processing, recycling, disposing, and monitoring of the waste materials. The costs of collection and transportation alone accounts for about 70% of the waste management costs (Tavares et al., 2009). These high collection and transportation costs combined with the forecast of increased waste production in cities demand a smart solution to deal with this problem.

The current method in which the vast majority of waste is collected is a static or periodic collection that is using fixed routes and is also revered to as *blind collection*. Here each truck follows specific routes on specific days and picks up all containers along its route, see Figure 1. This method often already includes some work of optimization, because some containers are emptied more often than others and routes are matched to a truck's capacity, but there is still a lot of room for improvement. Ramos et al. (2018) shows an example in which on average 10% of all bins on a route is empty, even reaching a maximum of 38% of the bins being empty. Furthermore, approximately 66% of the fill-levels was registered below or equal to 50%.

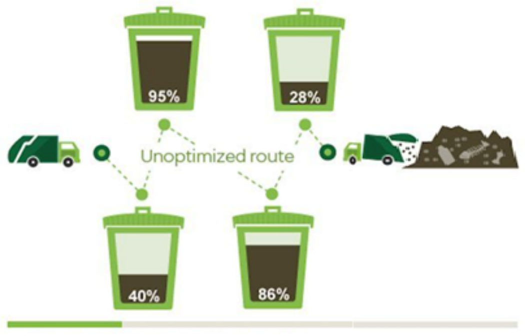

Figure 1: Unoptimized route
Waste-insight (2017)

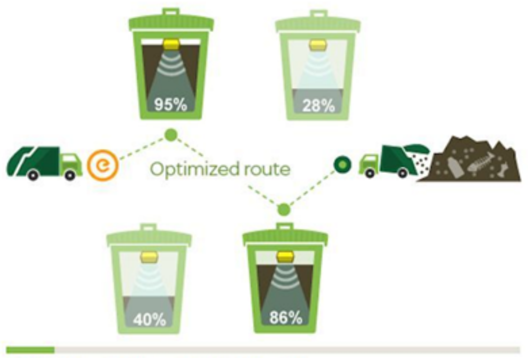

Figure 2: Optimized route
Waste-insight (2017)

This paper will discuss the topic of smart waste collection and routing problems. The basic idea of smart waste collection starts with the use and implementation of sensors into the waste containers to measure their fill-levels. Because these sensors can be integrated into the Internet of Things, (IoT) the acquired data-points can be sent to servers where they are stored, processed and used for forecasting, supervision, and finally making smart decisions to optimize the collection of waste. Figure 2 shows that when the fill-levels are known, not all containers need to visited and routes can be optimized. The optimisation is often based on a objective function, which generally minimizes the total cost and could include traveling costs, labour costs, truck-related expenses, penalties and maximizes profits obtained from the collected waste. The variables that are to be calculated based on the acquired data, are the number of trucks needed per collection day and which route each truck will follow at a specific moment. Figure 3 depicts the different sequential step in the management of smart waste collection.

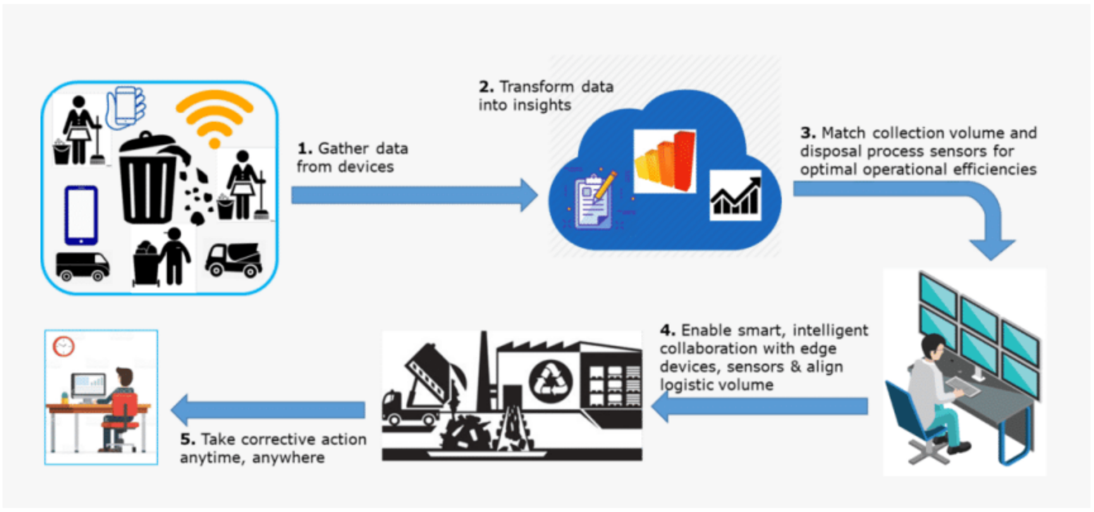
1. Gather data from devices
2. Transform data into insights
3. Match collection volume and disposal process sensors for optimal operational efficiencies
4. Enable smart, intelligent collaboration with edge devices, sensors & align logistic volume
5. Take corrective action anytime, anywhere

Figure 3: Steps in smart waste collection ORCA-Media (2019)

The process of smart waste collection can be optimized and solved in many different ways and on many levels of difficulty depending on the chosen system, chosen approach and chosen constraints. The simplest version of such an optimization problem is the Traveling Salesman Problem (TSP). A TSP is characterized by the use of only one vehicle, with a sufficing capacity to fulfil the demands of all customers (in this case the waste bins), which are known beforehand. A step up from the TSP is the Vehicle Routing Problem (VRP). The VRP is characterized by the use of a fleet of vehicles, which can have a limited capacity. The capacity is the strongest constraint and it determines how many customers each vehicle could visit before returning to the depot. All customers and their demands are again known beforehand. Notice that nor in the TSP's, nor in the VRP's basic formulation, time is considered what so ever. In both problems, the demand is set by the customers and the customers need to be known from the start. This is similar to the threshold method discussed in section 3.2. Due to the threshold constraint, all customers are known before a route optimisation is started, the only variable therefore is the route per vehicle and so both the TSP as the VRP have an objective of minimizing routing costs, -distance or -time. Having said this, both the TSP and the VRP have multiple variations in which time and other constraints can be introduced. A different type of routing problem is the Inventory Routing Problem (IRP). In this problem the inventory of customers is being managed, but in contrast with the TSP and VRP, the IRP is in charge of deciding via which route, at what time and how much inventory has to be delivered to which customer, preventing any stock-outs. This relates to the attractiveness method and the must-go may-go method in section 3.3 and 3.4, as these methods have the option of choosing the moment of collection as well. The IRP minimizes the sum of the inventory and routing costs.

In general, we could say that one could build an IRP out of multiple VRPs, one per day or unit of time. Depending on the chosen approach and time horizon for which a Smart Waste Collection Problem (SWCP) will be solved, a SWCP can be categorized as a VRP, as well as an IRP. The SWCP could be described as a revers IRP, as generally VRPs and IRPs deliver 'goods' to the customer, while in the SWCP the waste is collected from the customer and overflow should be prevented instead of stock-outs. To summarize, the IRP, attractiveness method and must-go may-go method, differ from the VRP and the threshold method in the way that the first three methods are in charge of customer selection. Meaning, the set of customers to visit each day is no longer given but will be determined based a certain conditions. Also the quantity is no longer set by the customer. For the SWCP it is slightly easier than for the IRP, since a collection means that instead of delivering goods, they are collected and instead of determining how much to deliver, the whole volume of the container is collected. To summarize, a VRP only decides which route to travel, a SWCP decides with route to travel and which customers to visit. Last, a IRP decides which route to travel, which customers to visit and how much inventory to deliver to each customer. However, vehicle capacity and time windows are used in this research, making it comparable to the CVRPTW (Capacitated Vehicle Routing Problem with Time Windows).

The growing population and urbanization forecast challenges for cities and urban areas. One of these challenges, the one this paper will focus on, is how to deal with the growing waste production in these urban areas. As technology improves, it allows optimization of already existing processes of which waste collection could be one. By using sensors to measure the fill level of waste containers an optimization program could be used to minimize overall costs, time, CO2 emissions. Although the foundation on which the WCP builds, (TSP, VRP and IRP) is thoroughly researched, the research into smart waste collection is actually quite new.

1.2 Research gap

In smart waste collection, sensors are used to measure the fill levels of waste containers. These measurements are used to decide which containers need to be collected on that specific day. Literature provides examples which show significant errors and uncertainties in these measurements. However, except for one article, no data treatment is discussed or proposed at all. Every paper assumes the data from the sensors can be used straight away. Another issue that has to do with the use of data is the type of data that is used for test cases. Test cases are used to prove the potential of a certain optimization method. The problem lies in the data that is used. This data is in most cases far from realistic, which makes the results unrealistic as well.

From literature it shows that the concept of smart waste collection is used on different types of containers and in different areas, using different types of container selection methods. Which container, area and container selection method provides the highest savings is hard to say as no comparison has been made yet. However, large containers appear to be more beneficial than the smaller ones, rural areas more beneficial than urban ones and combined use of threshold and attractiveness more beneficial than any of the two separate.

Table 1: Comparison between literature and this research with synthetic data (SD), Reality based data (RB), real data (R), threshold based collection (TH), attractiveness based collection (AC) and must-go may-go based collection (MGMG)

Paper	**SD**	**RB**	**R**	**TH**	**AT**	**MGMG**
Mes et al. (2014)		✓				✓
Ramos et al. (2018)			✓		✓	
Markov et al. (2016)		✓			✓	
Bueno-Delgado et al. (2019)	✓			✓		
Abdallah et al. (2019)			✓	✓		
Lozano et al. (2018)		✓		✓		
Akhtar et al. (2017)	✓			✓		
Hannan et al. (2018)	✓			✓		
This paper			✓	✓	✓	✓

1.3 Research questions

The aforementioned research gab has let to the formulation of the main research question. This will therefore be the main focus of this research. To structurally help answer the main research question, a set of sub questions was formulated as well.

Main research question:

How will different container selection methods impact the KPIs of the smart waste collection system, using a realistic data set?

Sub questions:
1: What are the container selection methods available in the literature and in practice?
2: What are the KPIs in a smart waste collection system?
3: How can the smart waste collection system be modeled for different methods of container selection?
4: What is the performance of different container selection methods based on the KPIs of the smart waste collection system?

1.4 Research approach

1.4.1 General approach

The double diamond method was originally created by the Design Council to reflect the design process. It was inspired on the design department of eleven big companies such as Microsoft, Starbucks, Sony and LEGO. Although this methodology was used for design processes, it helps to get a general overview of any project and the four faces of *discovering*, *defining*, *developing* and *delivering* appear in many different projects, one of them being a master thesis. The *discover* phase is in general used to gain insight in the problem. For this master thesis it will consist of the start of the literature research, with the aim of getting a good understanding of the subject. This phase is the beginning of the diamond and is about divergence. When the subject is well understood and a specific section is investigated in a more detailed way, the next phase is already underway. This is called the *define* phase. In the *define* phase the goal is to converge and focus on a specific area. This phase could end with defining a well formulated problem or research question. For this master thesis, this part will focus on finding the gab in literature and to formulate the main- and sub research questions. When the problem is formulated it is time to start to work on solutions. In the *develop* phase, the idea is to diverge again and consider all potential solution. This phase will be used to develop the methods proposed in this research. This phase is followed by the *deliver* phase, were the goal is to converge again to one solution. For this research this will mean testing the models, comparing the results and drawing conclusions from them.

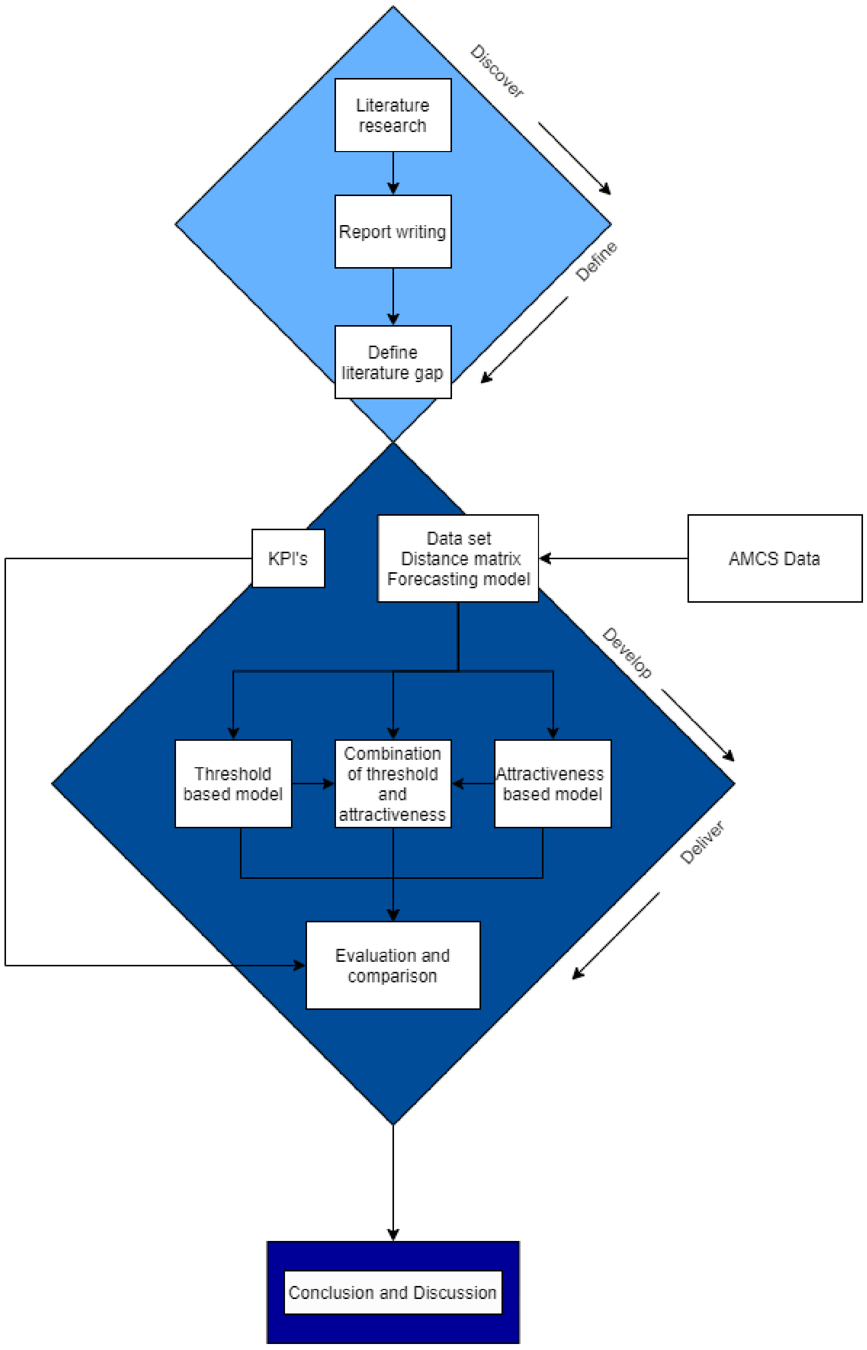

Figure 4: General research approach

1.4.2 Question specific approach

Sub question 1:
What are the container selection methods available in the literature and in practice?

To be able to answer sub question 1, both literature and practice need to be researched for the used container selection methods. After which a comparison can be done to see if literature and practice use similar methods. For literature, section 2 shows the articles that were read and compared and section 2.2 shows which container selection methods are used. In section 3.1 the comparison with AMCS is made and their used method is mentioned.

Sub question 2:
What are the KPI's in a smart waste collection system?

To determine the KPIs for a smart waste collection system, again both literature and practice need to be researched. For most literature articles the objective together with some constraints will show what the KPIs are for that specific smart waste collection problem. This can be found in section 2.4. As for practice, AMCS and OMRIN will be contacted to find their KPIs, their KPIs are discussed in section 3.1.

Sub question 3:
How can the smart waste collection system be modeled for different container selection methods?

This question requires both literature research and the acquired knowledge from doing the programming. First literature needs to be examined for different ways of modeling the smart waste collection system, after which positive aspects will used in the models of this research as well and other ideas will be adapted or removed. The insights from literature can be found in section 1.1 and the final results in section 3.2, 3.3 and 3.4.

Sub question 4:
What is the performance of different container selection methods based on the KPI's of the smart waste collection system?

Enable to answer this question many steps have to be taken. The main steps required to answer this question are: acquire data sets, read data sets into python, clean data sets, acquire distance and time matrix, make a simplistic forecasting model, write a pseudo code, make a mathematical model, build the model in python, test the model and finally evaluate the results, which can be found in section 4.2 and the conclusions in **??**.

Main research question:
How will different container selection methods impact the KPIs of the smart waste collection system, when compared using real data?

The main research question can be answered after individual evaluation of the container selection methods and comparing the results. When compared and evaluated additional tests can be done to investigate certain correlations. All results are shown in section 4.2 and the conclusions in section **??**.

2 Literature review

This section will go through literature on the topic of smart waste collection. The research papers will be compared on the way the gather and treat data, the multiple methods for container selection and the methods that are used to solve their specific smart waste collection problem.

2.1 Data

The optimization of routing problems is generally based on the number of trucks available and more importantly on the set of customers, i.e. containers that are selected for the collection. The set of containers ready for collection could in the future be based on the data coming from sensors, that measure and communicate the fill levels of all containers. This data is the foundation on which the route optimization will be built. Therefore, the level of success of route optimization depends on the accuracy of the data. This chapter will cover the topic of data for smart waste collection, from the way it is obtained to the way it is used for route optimization.

2.1.1 Sensors and data gathering

The fill level of waste containers can be obtained in many different ways, i.e. by many different sensors, each sensor measuring other physical properties. Each has different pros and cons. The most discussed type of sensor in literature on the topic of smart waste collection is the volumetric sensor. As the name suggests this sensor measures the volume of waste in the container, also called the fill level. It measures the height difference between the surface of the waste and the location of the sensor, often placed in the upper part of the container. Examples of volumetric sensors are capacitive, infrared, radar and ultrasonic sensors. Based on range, accuracy and angle of operation the ultrasonic sensor is often the product of choice (Abdallah et al., 2019; Lozano et al., 2018; Markov et al., 2016; Mes et al., 2014; Papalambrou et al., 2015). Other sensors which are mentioned in literature are load sensors, gas sensors and magnetic proximity sensors. These sensors measure respectively, the total weight of the waste in the container, the concentration of CO2 or the number of times the lid is opened. Regardless of the type of sensor that is used, the sensors measure a certain value which will always contain a level of uncertainty, created by different errors. The uncertainty is something which can not be eliminated, i.e. it will always be there. The errors that cause, or define the range of uncertainty can sometimes be minimized but never completely eliminated (NDT-ResourceCenter, 2006). Environmental conditions inside the container, e.g., dust, humidity and temperature, can strongly affect the accuracy, i.e the lack of errors, as well as the reliability of the sensor's measurement due to a large variety in material types and shapes Papalambrou et al. (2015). Data obtained from sensors should, therefore, contain two components, the numerical/measured value and the degree of uncertainty. Strictly, it should even be followed by the level of confidence but this is often neglected. It should be noted that neither errors nor uncertainties have anything to do with mistakes, data obtained via mistakes should be explained and excluded from the data set. In Mamun et al. (2016) an experiment is executed to find the errors in a set of ultrasonic sensor measurements. During 36 measurements the errors are found to be between -7.8% and +14.4% of the true fill level. Here 34 out of the 36 measurements, measured a fill level higher than the actual fill level, thus showing a systematic error. Furthermore, a trend is observed in the measurements that show a decrease in the magnitude of the errors with increasing fill levels. Factors such as the aforementioned should be taken into account to make the errors smaller and make the measurement more accurate as well as mistakes made in the measurements. Although most articles don't deal with uncertainties, data treatment or mistakes in the measurements, as will be further discussed in the next paragraph, some articles propose simple methods to reduce the errors and improve the accuracy of the obtained data. In the work of Papalambrou et al. (2015), a higher accuracy to cost ratio is obtained by the use of two sensors to measure the fill level of each bin.

The methodology of the use of two or more sensors is used by many articles such as Akhtar et al. (2017); Catania and Ventura (2014); Hannan et al. (2018); Johansson (2006). The use of LEDs is also suggested as the additional light enables the sensor to obtain more information about the area, height and shape of the waste inside the container. By this illumination, the volume estimation should increase in accuracy according to Johansson (2006).

2.1.2 Data treatment for optimization methods

As discussed in the previous section, data obtained from the sensors is not always reliable. Therefore it is wise to deal with these uncertainties and errors and try to find a way in which the reliability can be enhanced. This section will discuss the proposed ideas in literature on how to deal with the data coming from the sensors in the smart waste containers. Remarkably, in literature hardly any data treatment is proposed. Mes et al. (2014) are the only ones that do not directly propose using the measurements obtained by the sensors. They suggest to use estimates for the deposit volumes, the waste levels in the containers and the amount of waste overflow. Although the way these estimates are made is not described in their article, the purpose of the estimates is to cope with uncertainties in waste deposits. The lack of detail in this article about the way sensor data is treated and the pure absence of it in the rest of literature articles shows a discrepancy between academic work and reality, taking into account the lack of validation given for the direct use of the data.

2.1.3 Data generation for test cases

Due to the fact that installing hundreds or thousands of sensors is a time consuming and costly process, artificial data is often generated to be able to run test cases in order to prove the success of an optimization method for the WCP. Throughout the articles which are reviewed, a large variety of methods have used simulated data, one being more realistic than the other.

Using direct data from the sensors is most often not possible as aforementioned, although the data could be based on the actual fill levels of the containers in question. A good example of this is the article of Ramos et al. (2018), they have let the collection team track the fill levels of 3 routes for a time span of 30 days. Due to time windows in between pickups, the fill levels are averaged over this time window to obtain a daily deposit rate. This method captures the general deposit rates in the studied area but is not able to capture and use daily variations in the deposits. Abdallah et al. (2019) take it a step further. A similar approach is used, only now the fill levels are checked daily. The authors even include the field survey in the article, where the objectives were to understand the typical daily variations in the fill-levels of waste bins and use the collected data of single-family dwellings for the waste bins in the simulation. This is done by tracking the daily fill levels of 115 containers for different types of households (high-rise buildings, mid-rise buildings and single-family dwellings), in April in 2018. The month of April has been chosen, expecting it to represent regular data. By doing such an extensive field survey this article is able to use realistic data for their case study and actually determine how well their system would function in this area.

Not all articles need realistic data though, it depends on the goal of the article. When the goal is to test the proposed work in a test case, the data should give a realistic representation of the studied area. Mes et al. (2014), on the other hand, focuses on parameter tuning and for this reason chooses to make assumptions when generating data, consciously simplifying the tested case. They simplified the test case by using deterministic deposit volumes, but on the other hand, still uses stochastic inter-arrival times between deposits. The example of Mes et al. (2014) shows that simplifying the test case is acceptable when justified. The counterexample is Bueno-Delgado et al. (2019), here the fill levels are randomly set to levels between 1 and 100%. The lack of installed sensors is the only reason given for this assumption. This is an example of a generated data set in which more effort could have been made to justify the assumption or to generate a more realistic data set.

In the examples presented above, both ends of the 'realistic data' spectrum are discussed but many possibilities lie in the middle. Lozano et al. (2018) generates data based on statistics of selective waste production in local towns. The regional administration of Castilla y León was the source from which the historical data was derived. The same is done by Markov et al. (2016) who creates a set of IRPs, which are derived from real data obtained from the canton of Geneva, Switzerland. Instead of generating new data sets, already existing data sets can also be used. Akhtar et al. (2017) and Hannan et al. (2018) use renowned data sets which allow comparison to the work of others. The comparison of different types of heuristic can be made using these data sets of which many varieties exists. The drawback of these existing data sets is that these where also generated at some point in time and have certain imperfections such as randomly chosen container locations or the lack of a geographical information system (GIS). Waste deposit rates are randomly generated considering a mean waste generation rate with a standard deviation.

2.1.4 Summary

In smart waste collection, sensors are used to measure the filling levels of waste containers. These measurements are used to decide which containers need to be collected for that specific day. Literature provides examples which show that significant errors and uncertainties in these measurements. However, except for one article, no data treatment is discussed or proposed at all. Every paper assumes the data from the sensors can be used straight away. Another issue that has to do with the use of data is the type of data that is used for test cases. Test cases are used to prove the potential of a certain optimization method. The problem lies in the data that is used. This data is in most cases far from realistic, which makes the results unrealistic as well.

2.2 Containers

Waste containers come in all shapes, sizes and for many different types of waste such as fruit/vegetables and garden waste, general waste, metal, paper and plastic. In this chapter, we will focus on smart waste containers, i.e. a waste container fitted with a sensor that can measure the waste volume or weight and share the data so it can be used for optimization purposes. The technology to turn a 'normal' container into a smart one, can be applied to almost any container imaginable. Although it can be applied to any container does not mean it is useful to apply it everywhere. This chapter will discuss the different types of containers that are used, which containers and areas are most suitable for smart waste collection and finally, the procedure of container selection carried out by the different methods used in literature. The type of waste, collected per scenario is neglected as the same methodology can be applied regardless of the type of waste.

2.2.1 Types of containers and suitability

The most discussed containers in literature are the following three:

• Mini containers,	1 per household,	often 140 or 240 liters
• Block containers,	1 per large building or block,	varies between 500 to 5000 liters
• Underground containers,	1 per large building or block,	varies between 3000 to 5000 liters

Although the growing interest in smart waste collection is mainly driven by forecasted problems for cities in the future, this is not the only application for smart waste management. Looking at it from an optimization point of few, when many containers are placed relatively close together, i.e. in cities, the *damage* being done by collecting a container which is almost empty is relatively low. The damage being: additional costs, additional CO2 emissions, lost time, etc. This damage will significantly increase when travel distances between containers become larger, as in rural areas. Using this reasoning, Lozano et al. (2018) confirms multiple times that a smart waste collection management could be especially useful in rural areas. Quoting Lozano et al. (2018), "Journeys from one town in a region to another may sometimes be several kilometres long and skipping some towns may mean important savings on fuel and time over a year.". However, the economical capacities in rural areas are not efficient enough for the implementation. For this reason Lozano et al. (2018) plea for an energy-efficient, little maintenance and low-cost technologies to make smart waste collection interesting for rural areas. Mes et al. (2014) back up the argument of Lozano et al. (2018) and show that the highest gains are obtained in rural areas where the network is considered to have a size of 150 x 150 min driving time. Here the savings vary between 24% and 40%. In urban areas, however, with size 30 x 30 min, the saving are between 17% and 23%. This shows that although most papers in literature focus on urban areas, the highest savings can be realized in rural areas.

Smart waste collection is based on the data coming from the sensor and the decisions that can be made based on this data. Many papers use forecasting models to predict each daily or weakly waste deposit. When containers become smaller they become more vulnerable for perturbations in the behaviour of the deposit, making it harder to predict the ideal moment for collection and increasing the risk of overflows. Another, important decision variable for smart waste collection is the moment a container is emptied. This means that the trucks should be able to reach a container any time a day. Finally the smaller the container, the easier it is to empty them and the less damage is done when it appears to be empty. The three aforementioned arguments make mini containers and smaller city trash bins less attractive for smart waste collection. Larger containers have the benefit of being less vulnerable for perturbations in the deposit behaviour, being available for trucks to collect them 24/7 and cause higher damage when being emptied during collection. Underground and block containers are thus the most suitable candidates to be used for smart waste collection.

2.2.2 Container selection

As mentioned in the introduction of Chapter 2 the route optimization is based on the number of trucks available and more importantly, on the set of containers that are selected for pickup. The general idea of smart waste collection is the use of sensors, that collect and communicate the fill levels of all containers. Based on these fill levels, decisions are made about which containers and via which route they are collected. In the discussed literature, three main methods for container selection are proposed:

1. Based on a pre-determined threshold level of a container
2. Based on the attractiveness of a container
3. Based on both the threshold level and the attractiveness

Threshold based collection
The use of threshold levels is by far the easiest of all methods and most used in literature. However, a wide variety of it is used in literature. The simplest way is to set the threshold level equal to a percentage of the bin's capacity and collecting every container whose level is equal of higher than the threshold. Taking it one step further, Abdallah et al. (2019) and Lozano et al. (2018) include the predicted deposit volume for that day and add it to the fill level already present in the bin and measured by the sensor. Abdallah et al. (2019) does not mention the value of the threshold level or used safety margin and Lozano et al. (2018) obtained the used 80% threshold level, from a technical public tender document. However, Abdallah et al. (2019) does not investigate the optimal fill level but uses a preset threshold level to compare different deposit volumes and show the benefits compared to the conventional method of collection. More interesting and often done in literature is the optimization of the threshold level. Many articles include their own research in finding an optimal threshold level for which the total cost, costs/kg, amount of CO2 produced or amount of trucks is minimized. Bueno-Delgado et al. (2019) investigates these results for three different types of trucks, namely 1500kg, 2600kg and 6700kg. Their research shows a reduction of costs for the 6700kg truck up to threshold levels of 70%, above 70% the cost/kg exceeds the current path of the test case. Interesting is the fact that trucks of 2600kg only show a reduction in cost when fill levels are used between 50% and 70%. Akhtar et al. (2017) and Hannan et al. (2018) both optimize the threshold level using different data sets based on the tightness, i.e. the ratio of the total capacity of the used fleet over the total amount of weight collected. The optimum value found for the threshold level is between 70% and 75% for both articles.

Attractiveness based collection
Another method to select containers is based on the attractiveness of a container. Ramos et al. (2018) defines a container to be attractive when the fill level is maximized and the transportation costs incurred to collect them are minimized. Both factors are included in the optimization function as well as the penalty for the usage of a certain amount of vehicles. The objective function thus maximized the profit. In other words, it maximizes the amount of collected waste, the negative amount of distance travelled and the negative amount of trucks needed for the collection. Although Markov et al. (2016) does not use the word attractiveness, the method used in their article is closest related to the pickup decisions made, based on attractiveness. Using historical data, a statistical model is used to calculate estimate point demand forecasts for each day of the horizon including its error which is used to calculate the risk of overflowing or route failures. The objective function, that consists of three distinctive parts is minimizing each of the three parts and does not include a profit acquired from the amount of collected waste as done by Ramos et al. (2018).
Instead, Markov et al. (2016) uses an adaptive large neighbourhood search to find the moment for each container to be picked up for the smallest cost possible. The objective function consists of the Expected Overflow and Emergency Collection Cost (EOECC), the Routing Cost (RC) and the Expected Route Failure Cost (ERFC). The EOECC is used to capture the costs belonging to emergency collection and penalties for overflowing containers. The first happens when a sensor sends information about a container passing its maximum capacity and thus is starting to overflow. When this happens the container in question has to be collected during that day, possibly even when it is not included in any planned routes to be executed that day. The second is the penalty imposed by the municipality to the collector when any container overflows. The second part that the objective function is minimizing is the Routing Costs, which need little explanation. And thirdly the Expected Route Failure Cost, which are the additional costs required when the capacity of a collection vehicle is reached before completing its entire route. Using a set of operators, Markov et al. (2016) 'destroys' and 'repairs' single containers, vehicles, entire routes and even days. By destroying or removing containers, vehicles, etc. a newfound solution if found and it is compared with the previous one. Operators who found better solutions and thus lower objective functions are 'rewarded' by increasing their weights and are 'punished' if the solution did not improve.

Must-go may-go based collection
The last of the three methods is a combination of the other two. The only article that combines the two methods is Mes et al. (2014). Here containers are labelled as one of the following three categories: 'Must Go', 'May Go' and 'No Go'. The 'Must Go' containers are based on the first method, the one based on threshold levels. Interesting is the fact that Mes et al. (2014) do not set a threshold on the fill level of the container as other articles do (Abdallah et al., 2019; Lozano et al., 2018; Akhtar et al., 2017; Hannan et al., 2018). Instead, Mes et al. (2014) puts a threshold on the number of days in which a container is expected to be full. This threshold is included in one of the tunable parameters and can be different for different days of the week as waste is not collected during the weekends. 'May Go' containers are selected based on their attractiveness and are added to a route of 'Must Go's' if it is beneficial to do so. Here the attractiveness is defined as the insertion cost needed to add a 'May Go' container to a route based on 'Must Go's', divided by the estimated fill level of the container. However, this causes a more distant container to be less likely to be added compared to a container which is less distant. Mes et al. (2014) try to solve this problem by dividing the previous ratio by a historical smoothed average of it. In this way, each container's attractiveness is scaled individually instead of being compared to the attractiveness of others.

2.2.3 Summary

This chapter shows that the concept of smart waste collection is used on different types of containers and in different areas, using different types of container selection methods. Which container, area and container selection method provides the highest savings is hard to say as no comparison has been made yet. However, large containers appear to be more beneficial than the smaller ones, rural areas more beneficial than urban ones and combined use of threshold and attractiveness more beneficial than any of the two separate. Future research has to provide better insides in which method is better suited for a specific situation.

2.3 Optimization methods

Solving any variation of a VRP or IRP is a challenging task. Basic versions of the VRP are the TSP and the Chinese Postman Problem (CPP). The TSP can be categorised as an NP-hard problem, which means it can not be solved within polynomial time. The CPP, on the other hand, is not a NP-hard problem, only when capacity constraints are added and the problem becomes a capacitated-CPP, it can be classified as a NP-hard problem. NP-hard problems are difficult to solve and the goal is to find a good to an optimal solution within a relatively short amount of computational time (Santos et al., 2008). This due to the fact that the time between data transfer and waste collection should be kept within a short time span. Depending on the size of the container network, the extent to which stochastic waste deposits are used, the length of a planning horizon, the chosen constraints and many other factors, this challenging task is not suitable for every solution approach. Many authors use heuristics to solve these types of problem, other simplify by using assumptions, which allows again for exact solutions, but the assumption often makes the problem less realistic. Roughly speaking, a solution approach belongs or to the set of mathematical programming solutions or to the set of heuristics, with a few exceptions belonging to nether. This chapter will discuss some of the used methods in smart waste collection, both mathematical and heuristics.

2.3.1 Mathematical programming

Exact solutions, e.g. mathematical programming, produce accurate and optimal results. But these excellent results come at a price, they are not suitable for medium-large practical problems such as a complicated WCP (McLeod and Cherrett, 2008). The design of a mathematical solution approach is that it guarantees to find the optimal solution in a certain amount of time. The problem is that the computational time could exponentially increase with increasing dimensions of the problem that needs to be solved (Akhtar et al., 2017). Examples of methods that belong to the set of mathematical solutions are linear, mixed-integer, non-linear and dynamic programming. Bueno-Delgado et al. (2019) implements integer linear programming (ILP) in an open-source network planner, named Net2Plan. The formulation of the problem is written in such a way that it calculates the number of trucks that result in the shortest collection routes. These routes are assigned to specific trucks and the data of the city infrastructure is used via a GIS database. Both the number of trucks and their travel distance are minimized. A similar approach is used by Ramos et al. (2018), they also minimize the resources and the travel distance. Instead of ILP, a mixed-integer linear programming (MILP) is used, which limits only specific parts of the code to use integers. Ramos et al. (2018) combine their MILP with a heuristics that calculated at what time during the day the model should be run to be able to obtain a maximum profit. Heuristics often offer a quick and good solution to the problem when computational times of mathematical programming becomes too long.

2.3.2 Heuristics

Heuristics are often problem-dependent and therefor very specific. There are two types of heuristics that or most often used. The first type is construction heuristics, which find the final solution by interactions. The second type is descending heuristic, which tries to find the local optimum for a given solution. In contrast with heuristics, meta-heuristics are problem-independent. This means that meta-heuristics make few to no assumptions about the problem and therefore have large spaces of candidate solutions. Examples of meta-heuristic methods are ant colony optimization, local/neighbourhood search, tabu search and genetic algorithms. Some of these methods are not flexible enough for waste collection in real and smart cities, where these methods have to deal with different start and endpoint in routes, the maximum number of times streets can be crossed by a vehicle, the decision on the number of trucks to use and which routes to appoint to them (Bueno-Delgado et al., 2019).

Heuristics, meta-heuristics and combinations of the two are all used for solving the WCP. The goal of Mes et al. (2014) is to develop a heuristic for the WCP, or IRP as they call it. This heuristic should specifically be fast (the goal of every heuristic) and parameterized, the latter also including a methodology to determine the best parameter settings using optimal learning techniques. While Mes et al. (2014) is using a single heuristic, Lozano et al. (2018) is using different heuristics and local search methods. All these heuristics and local search methods compete to find the best solution within 24 hours, which likely guarantees good results but is quite a time consuming for a WCP that relies on real-time data. Markov et al. (2016), on the other hand, rely on a single meta-heuristics. They use an adaptive large neighbourhood search (ALNS) algorithm, where the total cost is minimized over a rolling time horizon. Their objective function includes penalties for overflow, additional costs for route failures, costs for emergency collection and finally routing costs that also include a time component. The ALNS uses many destroy and repair operators, that change current solutions up to 30-40%. The success rates of the operators are continuously updated and influence the next iteration. Another meta-heuristic is used by Akhtar et al. (2017). They use a backtracking search algorithm (BSA), which is a population-based meta-heuristics and is relatively new. In contrast with Mes et al. (2014), this meta-heuristic only has one control parameter, which simplifies the algorithm. Finally, Hannan et al. (2018) also uses a population-based algorithm but they use a particle swarm optimization (PSO) algorithm. This PSO starts with generating initial particles by using a sweep algorithm, after which particles are dragged towards the optimal solution with a randomly generated velocity. The algorithm keeps track of the two best solutions per iteration.

2.3.3 Summary

As routing problems developed from simple travelling salesman problems to the complex vehicle- and inventory routing problem, methods for solving them also developed. Where heuristics provide good to optimal solutions, mathematical programming guarantees to find the optimum solution. However this comes at a price, mathematical programming includes large computational times. This makes them unsuitable for complex problems or problems that need to be solved in a relatively short amount of time, such as the waste collection problem. Heuristics offer quick and good solutions making them better suitable for the waste collection problem. Nowadays, often meta-heuristics are used or even multiple heuristics of meta-heuristics at the same time. In the latter, these heuristics compete for the best solution within a set time limit.

2.4 Synthesis of the literature

Is smart waste collection as profitable and efficient as it appears to be? And if so, what are the trends and most promising, realistic methods for optimizing the WCP. This chapter will be the foundation up on which the discussion and conclusion will build to answer the questions. This chapter will discuss the most relevant articles and is organised per article. It will show the details and specifications per article.

2.4.1 Results per article

Mes et al. (2014), Inventory routing for dynamic waste collection.

Specifications per article	
Type of sensor:	Volumetric
Type of container:	Not specified
Type of container selection:	Must-Go May-Go
Optimization method:	Mathematical (MILP)
Area of focus:	Both rural and urban (simulated)

Results:
By optimization of the tuning parameters the weighted costs per unit of volume are decreased by 40% compared to the default settings. Default settings perform reasonably well in other networks. When more realistic networks are considered, where more variables are day dependent, savings can be higher than 40%.

Note: The maximum saving of 40% was obtained in one of ten test cases. The average saving was 23.8% with a minimum in one test case of 16%. Especially the large virtual networks scored above average, and thus better than the small virtual networks. The larger ones correspond to a province while the smaller ones correspond to urban areas. This appears to be evidence that larger-scale operation would be more beneficial than smaller ones.

Ramos et al. (2018), The smart waste collection routing problem: Alternative operational management approaches.

Specifications per article	
Type of sensor:	Ultrasonic
Type of container:	Block container
Type of container selection:	Both threshold and attractiveness
Optimization method:	Parameterized heuristic
Area of focus:	Provincial

Results:
The third method that is proposed has the most potential and shows the highest savings compared to the current way of working. A potential profit increase of 7% is found. The kg/km ratio is increased by 19%, distance travelled is reduced by 33% and the vehicle usage rate stays approximately the same.

Note:
In the third scenario, an overflow percentage of 1% is allowed, with a maximum overflow of 20% of the bin's capacity. Furthermore, the daily deposit rate was determined by dividing the total amount of collected waste by the 30 days in which it was collected. Any trends or day timedependent variations are removed in this way, making it less realistic. What adds to this is that distances between containers were calculated using Euclidean distance, but were corrected with a factor to adjust for roads not being straight lines.

Markov et al. (2016), Inventory routing with non-stationary stochastic demands.

Specifications per article

Type of sensor:	Ultrasonic
Type of container:	Block container
Type of container selection:	Attractiveness
Optimization method:	Meta heuristic (ALNS)
Area of focus:	Urban

Results:
Although the algorithm of Markov et al. (2016) performs very well when compared to IRP and VRP benchmarks from literature, the results of the test case is quite disappointing. Computational times of their algorithm are good and the problem can be solved in around 10 to 15 minutes and the amount of collected waste is on average more than twice as many containers. But the costs of solving the WCP for their complete objective function in terms of expected cost are 50 to 60% higher, compared to the routing-only. Which is strongly influenced by the emergency collection cost. Not all increase in cost is caused to the emergency collection cost when the complete objective function is compared to the routing-only an increase in the cost of 30 to 35% is observed. The difference in cost between the percentages above is due to the expected overflow cost. This also shows that the route failure cost is practically zero in both situations. Apparently, including probabilistic information and forecasting in the objective function led to an increase in cost.

Note:
Although this article comes closest to capture a realistic version of the WCP their result is not very promising at this moment. A comparison with the current way of the waste collection could have provided a reference for the routing-only solution as well. As mentioned in the article, when more accurate forecasting models are used, the gap could be reduced significantly.

Bueno-Delgado et al. (2019), Optimal path planning for selective waste collection in smart cities.

Specifications per article

Type of sensor:	Not specified
Type of container:	Block container
Type of container selection:	Threshold
Optimization method:	Algorithm (Net2Plan-GIS)
Area of focus:	Urban

Results:
In the comparison, the influence of multiple truck sizes and threshold levels on the average waste collected in kilograms, the number of trucks that are needed, the CO2 emissions in kilograms and finally the total cost in /kg, is investigated. The results show that the larger the truck the less of an influence the threshold level is. Large trucks (6700 kg) have lower costs than the current way of collection between 0 and 70% threshold levels. The higher the threshold level the lower the CO2 emission, which is almost equal for all truck sizes. Smaller trucks (2600 kg) can also decrease the cost in comparison to the current way of waste collection, but it is limited to the threshold range of 50 to 70%. Also, a single truck is no longer sufficient, while two truck is not used to full capacity.

Note:
Although the methodology of this article is admirable, the way waste deposit rates and volumes are used is not even close to realistic. Container volumes are randomly generated each day with a fill level between 0-100%. Using realistic deposit rates and volumes would make a much more interesting article.

Abdallah et al. (2019), Simulation and optimization of dynamic waste collection routes.

Specifications per article	
Type of sensor:	Ultrasonic
Type of container:	Block container
Type of container selection:	Threshold
Optimization method:	Algorithm (Decision making)
Area of focus:	Urban

Result:
The shortest path algorithm for route optimization, GIS based, acquired a operational cost reduction of 19% when compared to the current way of waste collection. By investigating the reduction of fill rate, a decrease of 40% in fill rate showed a decrease in operational cost of only 25%. Showing that the amount of waste is not directly linked to the operational costs.

Note:
Because 88% of the population are ex-pats and travels back to home countries, also lower waste deposit rates, 90, 80, 70 and 60% where used. This makes the studied area very susceptible to drastic changes in waste deposits. This could be an interesting follow-up study, to see if such drastic changes can be managed and what would be the differences between current and smart waste collection.

Lozano et al. (2018), Smart waste collection system with low consumption LoRaWAN nodes and route optimization.

Specifications per article	
Type of sensor:	Ultrasonic
Type of container:	Block container
Type of container selection:	Threshold
Optimization method:	Multiple (meta-)heuristics
Area of focus:	Provincial

Results:
The developed sensor can be in operation for more than a year and measure weight, temperature and volume of the container. The coverage of the area by antennas also proved to be feasible. Route optimization resulted in an average saving of 28% in terms of travelled distance. These savings will create financial savings due to saving of time, truck usage and workforce costs.

Note:
The results show a clear pattern in the length of collection routes and thus in waste deposits. It is surprising that such a pattern was not noticed by the collection company. The result would have been much smaller when the collection company had alternated between long and short routes each day.

Akhtar et al. (2017), Backtracking search algorithm in CVRP models for efficient solid waste collection and route optimization.

Specifications per article	
Type of sensor:	Multiple
Type of container:	Not specified
Type of container selection:	Threshold
Optimization method:	Meta-heuristic (BSA)
Area of focus:	Not specified

Results:
By using a BSA in a CVRP a distance reduction of 36.80 % was observed. This will lead to reduction of fuel consumption and CO2 emissions by 50% and 44.68%, respectively.

Note:
These results look promising, but a few side notes have to be added. First of all the time span in which waste was collected and on which the results are based is to short. By using only a single day of collection and later on a week from Monday to Monday, trends and uncollected waste are neglected i.e. the start-up period is not finished and no stable state has been reached. With longer time spans these trends would have been captured.

Hannan et al. (2018), Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm.

Specifications per article	
Type of sensor:	Multiple
Type of container:	Not specified
Type of container selection:	Threshold
Optimization method:	Meta-heuristic (PSO)
Area of focus:	Not specified

Result:
By using a PSO algorithm and a number of renowned data sets, a 70-75% threshold level is found to create the highest savings.

Note:
As being written by a co-author of Akhtar et al. (2017) and being published after the aforementioned article, the result is exactly the same. The only new thing here is that a PSO is used instead of a BSA.

2.4.2 Summary results

Table 2: Summary of articles discussed in Chapter 5

Authors	Sensor	Container selection	Type of container	Optimization method	Area of focus
Mes et al. (2014)	Ultrasonic	Threshold + Attractiveness	Not specified	Heuristic	Rural and Urban
Ramos et al. (2018)	Volumetric	Attractiveness	Block container	Heuristic + Mathematical (MILP)	Provincial
Markov et al. (2016)	Ultrasonic	Attractiveness	Block container	Meta-heuristic	Urban
Bueno-Delgado et al. (2019)	Ultrasonic	Threshold	Block container	Mathematical (ILP)	Urban
Abdallah et al. (2019)	Ultrasonic	Threshold	Block container	Not specified	Urban
Lozano et al. (2018)	Ultrasonic	Threshold	Block container	Multiple heuristics	Provincial
Akhtar et al. (2017)	Multiple	Threshold	Not specified	Meta-heuristic	Not specified
Hannan et al. (2018)	Multiple	Threshold	Not specified	Meta-heuristic	Not specified

3 Methodology

In this section the basic concepts, details and mathematical models of all three container selection models are presented. The verification and validation of each model is also included in this section.

3.1 Conceptual model

As explained in section 1, the SWCP can be both categorized as a VRP or an IRP. This solely depends on the selection of customers. When it is known which customers to visit on which day this model will be categorized as a VRP. When it still has to be decided when to visit each customer, it is categorized as an IRP. For each model, the goal is the same. Namely, optimize the objective function, i.e. minimize total travel time, and stay within an acceptable amount of overflows during the 12 day test period. All three models discussed in this research will have many things in common. All of the models have common indices, sets, parameters, a common objective function and common constraints. sections 3.2, 3.3 and 3.4 will go into the specifics.

3.1.1 Comparison to AMCS and OMRIN

From literature, it became clear that not many articles use real data or real scenarios, although statements were made about how their models would perform under real circumstances. This was the main motivation to get the models in this research as realistic as possible within the available time frame. Some details were left out, but overall the models posses many real features copied from or based on the AMCS models and information gained from OMRIN, which are successfully used in practice. The following details are also used by AMCS and OMRIN:

- The use of the must-go may-go method
- Real fill levels for the starting day
- Real daily deposit rates
  (except for some due to collection in real- time)
- Real network of containers, terminal and depot, including locations and volumes
- Realistic travel times calculated using the actual road network
- Same time windows for a subset of containers in the city centre
- Same amount of trucks/routes
- Same volumetric truck capacity
- Same starting times for the trucks
- Same service times for the containers and the depot

Some features were to time-consuming to include in the models, or unnecessary to include. The models described in this research differ by the following details from AMCS and OMRIN:

- Breaks at the terminal from 12:45 to 13:00

  In this model, all routes start at the terminal, visit the depot and end at the terminal. In reality, the first routes of the day start at the terminal, visit a depot and from here a new route will start. Meaning in reality trucks do not always go back to the terminal in between routes. However, the trucks of OMRIN do go to the terminal for a 45 minute lunch break. The assumption is that the additional travelled time by ending each route at the depot will compensate for the time needed to drive to the terminal for the lunch break.

- Additional orders besides the 408 containers

  Besides the 408 containers considered in the Leeuwarden area, also custom orders can be added. These are additional waste collections, e.g. collect extra waste at a company.

- Visits to Harlingen twice a week, with 1 truck

  The models in this research contain two trucks which can both do two routes. A detail which was not included is that twice a week one of the two trucks used will visit Harlingen to collect containers. Such detail would, of course, be possible to integrate, but due to time limitations was ignored.

- An additional depot outside Leeuwarden

  The above-mentioned reasoning holds as well for a second depot. Since Leeuwarden is the area of focus, not only the visit of Harlingen but also the additional depot was left out.

- Limited collections on Saturday and no collections on Sunday

  Since most people do not work during the weekend, the collection is minimized in weekends as well. On Saturday, only 1 truck does one route of necessary collections. On Sunday, no collection is done at all. This detail is not applied to the models of this research, as collection is performed every day.

- Smaller network

  The initial goal of this research was to compare the designed models also with the results from AMCS. The whole network of 408 nodes would therefore be used in the models. Unfortunately, the computational time for solving the threshold model with 408 nodes passed the time of 100 hours without getting close to a result. Since the threshold model is the simplest model to solve, and therefore needs the smallest computational time in order to be solved the other models had to be scaled down as well.

- No deposits between measurement and collection

  Since the data is recorded around 6:00 AM and the trucks will start there route at 6:45 AM, it is possible that waste will be deposited in the meantime. This is almost guaranteed for containers that will be collected in the afternoon. In reality, this could lead to route failures. A route failure occurs when the maximum capacity of the vehicle is reached before it finished its planned route. In such a case the vehicle has to first visit the depot before it can continue and finish its old route, which increases the total driving time considerably. Since in this research no real collection is done, the collected waste volumes will always match the predicted waste volumes. When the models of this research would be tested in reality, then an appropriate buffer could be taken on the truck's capacity to prevent route failures. Note, both the threshold model and the must-go may-go model have built in buffers on top of there threshold levels.

3.1.2 Key performance indicators

In order to compare the performance of the different models, some indicators have to be selected. In the case of the SWCP, there is a wide variety of KPIs being used in literature and practice. The performance of the SWCP can be evaluated based on minimizing travelling distance, travelling time, CO2 emission, total cost, the number of vehicles used, the amount over overflows or maximizing the vehicle capacity rate or the amount of collected waste. The most common KPI used in literature is by far a form of cost. An objective function for cost can be as simple assigning some monetary value to the number of driven kilometres or to the time taken for collection. It can also be very complex, as the objective function of Markov et al. (2016). Most of the other variables can easily be transformed to be included in a cost objective function. However, total travel time is chosen as the main KPI for this research. The main reason being that it is also the KPI used by AMCS and OMRIN, therefor being able to compare results. Although other KPIs are not explicitly incorporated in the objective function, they are very much related to the total travelled time. It is easy to understand that by minimizing the total travelled time, travelled distance is also roughly minimized and therefore as well to the emitted amount of CO2. Collecting the same amount of containers with multiple trucks will always result in higher total travel times, and therefore will be avoided, since each truck has to start at the terminal, visit the depot and end at the terminal again. The usage of fewer trucks will also translate into an increase in truck capacity utilization. Another KPI being maximized by minimizing travelled time is the number of overflows. When no threshold or upper limit is set, containers will never be collected. Therefore a negative relation arises between the total travelled time and number of overflows. The amount of overflows will be monitored for all test runs and a baseline will be determined, using the number of overflows present in the data of AMCS and OMRIN. To conclude, **total travel time** will be the main KPI while the **amount of overflows** is kept close to the amount observed in the used data. Note that the total travel time is specified in the objective function of each model as being the summation of all travel times, plus the summation over all service times.

3.2 Threshold based model

This section will discuss the threshold model. This threshold based model uses container specific threshold levels for container selection. Container specific threshold levels mean that containers are divided into groups, depending on their average daily deposit rate. If the average deposit rate is low, only small deposits are made and this container can be collected at a high fill level without risking large overflows. When a container has a large average daily deposit rate, this means that a container could be completely filled in only two or three days. This will result in a low threshold level. After the selection of containers, which only for this model can be carried out separately from the route optimization, the route optimization performed to find the smallest total travel time possible, in which these containers can be collected by two trucks with a maximum of two routes per truck per day. Figure 5 shows how the threshold model progresses from day to day. For each day the container selection is performed based on the threshold levels, after which the optimal route can directly be executed. No (rolling) horizon is used as in contrast with Figure 7. This means that no future information is used during the container selection and route optimization.

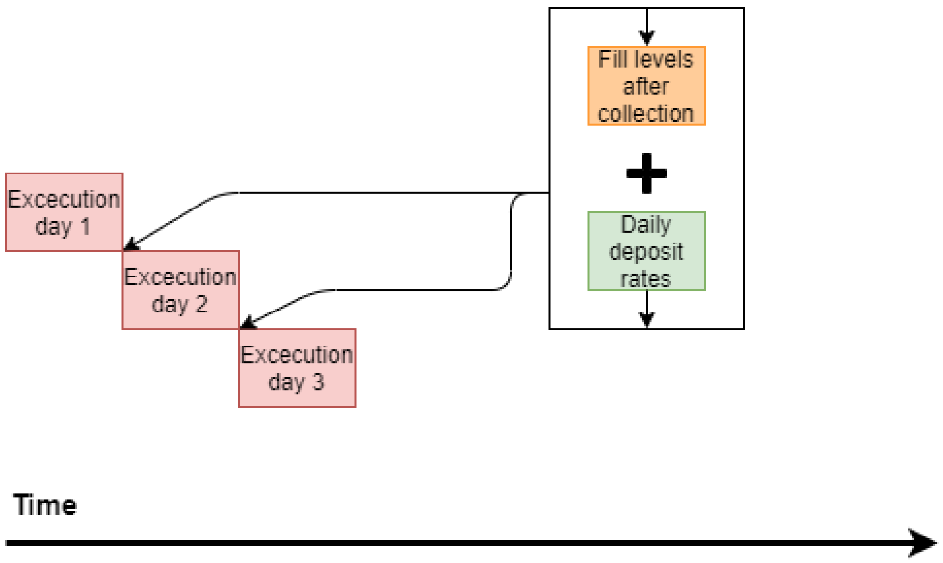

Figure 5: Visualisation of threshold based method

3.2.1 Differences from literature

Based on literature research, certain decisions were made when developing this model. The goal was to learn from literature, reject the bad ideas such as random fill levels and use the ones with potential such as a rolling horizon approach. An important difference from literature is that this research was carried out, using real data, obtained from smart waste container that are already in use. Many articles use generated data and in most cases even unrealistic data such as Bueno-Delgado et al. (2019). By using the real fill levels on the first day and daily deposit rates after this, this research is able to capture both a real starting point and a real deposit rates. Besides the usages of real data, this research also acknowledges the fact that real data contains errors. Therefore the data is screened, different errors where observed and dealt with if necessary. As for the model itself, the threshold model uses container specific threshold levels to distinguish between fast and slow running containers. Still, many articles use one threshold level for all containers like Akhtar et al. (2017) and Hannan et al. (2018). Limiting there possibilities for optimization. The threshold levels in this research are also considering the deposits that could be made in between the measurement and the moment of collection, following the example of Abdallah et al. (2019) and Lozano et al. (2018). Therefore providing a buffer to prevent overflows.

3.2.2 Mathematical model

Indices and sets:

i :	index for each node in the network.	$i \in ned, nid, early\ or\ visit$
j :	index for each node in the network.	$i \in ned, nid, early\ or\ visit$
k :	index for the number of routes.	$k \in K$
ned :	set of nodes excluding depot and terminal	$[0, ..., 406]$
nid :	set of nodes including depot(408) and terminal (407, 409)	$[0, ..., 409]$
$early$:	set of nodes that need to be collected before 9:00 AM	$[23, 57, 71, 72, ...]$
$visit$:	set of nodes which need to be collected	$[..., ..., ...]$
K :	set for the number of routes needed	$[0, ..., total\ volume/Q]$

Decision variables:

$x_{i,j,k}$:	Binary variable which is 1 if arc i->j is used by vehicle k and 0 otherwise.
$se_{i,k}$:	Integer variable which is 1 for the first node, 2 for the second etc.
$w_{i,k}$:	Continues variable which equals the collected volume of truck k at node i.
$u_{i,k}$:	Integer variable which equals the time a node is visited, lower bound = 24300. Starting time = 24300 sec => 6:45 AM.

Parameters:

$time\_matrix_{i,j}$:	Travel time between node i and node j, in seconds.
$threshold\_level_i$:	Threshold levels for node i, in percentages.
$container\_volumes_i$:	Total volume of the container for node i, in liters.
$fill\_level\_list_i$:	Fill level of node i, in percentages.
st_i:	Service time of node i, in seconds.
Q :	Maximum load of each vehicle.
T :	Time before all containers in early need to be collected.

Objective function:

$$\text{MIN} \sum_{i \in nid} \sum_{j \in nid} \sum_{k \in K} \text{x}_{i,j,k} * \text{time\_matrix}_{i,j} + \text{st}_i * \text{x}_{i,j,k}$$

Constraints:

In ascending order, constraint (1) ensures that all nodes that should be visited will be visited. Constraint (2) will prevent arcs from and to the same node. Constraints (3-7) forces each route to start at node 407, visits node 408 and ends at node 409. Note that node 407 and 409 are two nodes representing the same terminal and node 408 is the deposit site. Constraint (8) ensures flow conservation at each node. Constraint (9) prevents nodes to be visited that should not be visited. Constraints (10-14) prevent sub tours by numbering of nodes and only allowing the visit of nodes with higher numbers. Constraint (15) sets the load at start point equal to 0. Constraints (16 and 17) ensure load continuity and constraint (18) limits the load of each vehicle. Constrains (19 and 20) ensure visit time continuity and finally constraint (21) makes sure all nodes in early are collected before 9.00AM. Note that in constraints (16), (17), (19) and (20) the big M method is used. The M displayed in the constraint represents the number 100000.

$$\sum_{j \in nid} \sum_{k \in K} x_{i,j,k} = 1 \qquad \forall i \in visit \tag{1}$$

$$x_{i,i,k} = 0 \qquad \forall i \in nid, \forall k \in K \tag{2}$$

$$\sum_{j \in nid} x_{terminal,j,k} = 1 \qquad \forall k \in K, \tag{3}$$

$$\sum_{i \in nid} x_{i,terminal,k} = 0 \qquad \forall k \in K, \tag{4}$$

$$\sum_{i \in nid} x_{i,depot,k} = 1 \qquad \forall k \in K \tag{5}$$

$$x_{depot,terminal,k} = 1 \qquad \forall k \in K \tag{6}$$

$$\sum_{j \in nid} x_{terminal,j,k} = 0 \qquad \forall k \in K \tag{7}$$

$$\sum_{j \in nid} x_{i,j,k} - \sum_{j \in nid} x_{j,i,k} = 0 \qquad \forall k \in K, \forall i \in nid \tag{8}$$

$$x_{i,j,k} = 0 \qquad \forall i \in nid, \forall j \in mainlist, \forall k \in K \tag{9}$$

$$se_{terminal,k} = 1 \qquad \forall k \in K \tag{10}$$

$$se_{i,k} \geq 2 \qquad \forall i \in nid, \forall k \in K \tag{11}$$

$$se_{i,k} \leq |nid| \qquad \forall i \in nid \setminus \text{terminal}, \forall k \in K \tag{12}$$

$$se_{i,k} - se_{j,k} + |nid| * x_{i,j,k} \leq (|nid| - 1) \qquad \forall i \in nid \setminus \text{|nid|-1}, \forall j \in nid \setminus \text{terminal} \tag{13}$$

$$se_{i,k} - se_{j,k} + |nid| * (1 - x_{i,j,k}) \geq -1 \qquad \forall i \in nid \setminus \text{|nid|-1}, \forall j \in nid \setminus \text{terminal} \tag{14}$$

$$w_{terminal,k} = 0 \qquad \forall k \in K, \forall day \in days \tag{15}$$

$$w_{j,k} \geq w_{i,k} + cc_j - M * (1 - x_{i,j,k}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{16}$$

$$w_{j,k} \leq w_{i,k} + cc_j + M * (1 - x_{i,j,k}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{17}$$

$$w_{i,k} \leq Q \qquad \forall i \in nid, \forall k \in K \tag{18}$$

$$u_{j,k} \geq u_{i,k} + st_i + time\_matrix_{i,j} - M * (1 - x_{i,j,k}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{19}$$

$$u_{j,k} \leq u_{i,k} + st_i + time\_matrix_{i,j} + M * (1 - x_{i,j,k}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{20}$$

$$u_{i,k} = T \qquad \forall i \in early, \forall k \in K \tag{21}$$

3.2.3 Model optimization

To optimize the threshold model, only one parameter can be changed that will influence the selection of containers. Since this model already uses container specific threshold levels, the question arises '*how to change all these threshold levels to optimize the total travel time for the whole system?*'. This will be done using a threshold buffer to prevent overflows. The threshold buffer is the difference between a full container (100%) and the summation of the measured fill level and the forecasted average deposit rate. Since almost all forecasting models have some error, it is common to have a buffer larger than zero to avoid many overflows. Note that this research will also use negative threshold buffers to increase the number of overflows. Below, figure 6 is a small piece of Python code and it shows an example of how this threshold buffer is defined and used. For containers that have an average daily deposit rate is between 0 and 5 percent, the threshold level is set to 90%. This means that when around 6:00AM one of these container with an average daily deposit rate between 0 and 5 percent, is measured to have a fill level of 90% or more, the forecasted deposit rate, it was to be collected that day. This also implies that if the measured fill level was exactly 90% and the forecasted maximum of 5% was deposited before collection, a buffer of 5% would remain as a buffer. The same is done for average daily deposit rates between 5 and 10% and so on. In short, the threshold buffer is the buffer to prevent overflows, which is the same for all containers, regardless their threshold level.

```
threshold_levels = [ ]
for x in average_dailydepositrate:
    if x ≥ 0 and x ≤ 5:
        i = 90
    if x ≥ 5 and x ≤ 10:
        i = 85
    if x > 10 and x ≤ 15:
        i = 80
    if x > 15 and x ≤ 20:
        i = 75
    if x > 20 and x ≤ 25:
        i = 70
    if x > 25 and x ≤ 30:
        i = 65
    if x > 30 and x ≤ 35:
        i = 60
    if x > 35:
        i = 55
    threshold_level.append(i)
```

Figure 6: Threshold buffer

Since the buffer influences all fill levels it also has an influence on the KPIs, which are the number of overflows and on the total traveling time. When the value of the threshold buffer is increased, a higher buffer is created, minimizing the number of overflows. On the other hand a larger buffer will limit the system more, since containers need to be collected earlier. The entire struggle of waste collection is the trade off between minimizing overflows and minimal travel time.

To conclude, the threshold model has only one tuning parameters:

- **Threshold buffer**

3.2.4 Verification

The verification of a model is the step where the researcher tests to what extend the conceptual model has been correctly transferred into a computerized model. Besides the debugging of the code this means checking if generated outputs will match the expected outputs. Examples of such tests are event tracing, continuity tests, degeneracy tests, consistency tests and fault injection.

In this subsections the verification of the threshold model will be discussed and visualised. Further on in this report also the two other models will be verified. Each model is verified with the 10 tests listed below. Each test was performed using the same data set, and only critical parameters where added, removed or adjusted.

Continuity test: runs with slightly different parameters.
1) Basic run, with all constraints to test if the result is logic (bigger than 0).

Event tracing: check event order, causal relations, event times
2) Does the total time equals the service times and travel times.
3) Checking if order of visited nodes is in order and is equal to total visited nodes.

Degeneracy test: extreme cases.
4) Test without vehicle capacity constraint (only one vehicle should be used).
5) Test with vehicles capacity at 0.
6) Test with fill levels = 0 (no container should be collected)
7) Test with fill levels at 110 (all should be collected.
8) A test run where we remove the threshold level constraints, no container should be collected.

Consistency checks; e.g., doubled capacity à halved utilization.
9) Half the vehicles, double their capacity.

Fault injection: check whether faulty input is detected by the model.
10) Run without vehicles.

The table below contains the results of the verification tests. It describes the tested condition, the expected result, the obtained result and if the verification test was passed. This test was ran, using the AMCS data of all 408 containers on 14=06-2020. From left to right it shows what is tested, what the expected result/behaviour is and what the actually measured result was. If the expected result meets the measured result, the test is passed.

Table 3: Verification results threshold model

Test	Expected results	Obtained result	Pass/Fail
1) Normal conditions/full model	Total time >0 Total volume > 0 Nr. of containers > 0 K >0	Total time = 33436 s Total volume = 249911 L Nr. of containers = 74 K = 3	Pass
2) Check event order	se[terminal, k] = 1 se[j, k] - se[i, k] = 1	se[terminal, k]=1 se[j, k] - se[i, k] = 1	Pass
3) Total time check	Travel + service time = 33436s	Travel + service time = 33436s	Pass
4) No vehicle capacity limit	K = 1 Nr. of containers = 74	K = 1 Nr. of containers = 74	Pass
5) Vehicle capacity at 0	Infeasible	Infeasible	Pass
6) All fill levels are 0	Total time = 0 K = 0	Total time = 0 K = 0	Pass
7) All containers are in early	All collected before 9:00AM Or infeasible	ll collected before 9:00AM	Pass
8) No threshold level constraint	Total time = 0	Total time = 0	Pass
9) Double truck capacity	K = 2	K = 2	Pass
10) Run without vehicles	Infeasible	Infeasible	Pass

3.2.5 Validation

The validation of a model is the step where the researcher tests to what extend the computerized model represents the real system. Section 3.1 describes all the simplifications and details applied in the models of this research. As already mentioned, a full case study and comparison to AMCS was not possible. This limits the option to compare the model to reality and that way validate the models discussed in this research. However, all total driving times of AMCS for the test period are known, as well as the number of nodes visited. These can be used to see if the models presented in this research, provide solutions in the same order of magnitude as the solutions of AMCS. AMCS visited in the 12 days of the test period, 1006 nodes in a total travel time of 569580 seconds. However, the AMCS network consists of 408 containers and the one used in this research 151. The table below shows the comparison between the AMCS solution and the solutions of the threshold model.

Table 4: Validation results threshold model

	Total travel time [s]	Nr. of nodes visited
AMCS	569580	1006
AMCS Scaled	210800	416
Threshold 10%	158717	328
Threshold 5%	156354	318
Threshold -1%	147097	298

Firstly, Table 4 shows that the results of this research are in the same order of magnitude as the results of a model which is successfully applied in real life. However, AMCS shows to have a higher *scaled total travel time* and a higher *scaled number of visited nodes*. Both of which are not necessarily positive. Visiting more nodes leads to higher total travel times, which for both AMCS and this research is the minimized KPI. It is, however, important to note that AMCS also schedules custom orders and has to deal with more regulations than the models of this research and visits the city of Harlingen. As already mentioned, a comparison with AMCS at this point is very hard and therefore only used to show that the results are of the right order of magnitude.

3.3 Attractiveness based model

This section will discuss the attractiveness based model. The idea behind an attractiveness model is to collect containers at the most attractive point in time. However, attractiveness is an opinion. It can differ from person to person or from company to company. Some articles and companies focus on minimizing cost, others on minimizing overflows and other on minimizing time. Since AMCS and OMRIN use total travel time as an KPI, this will also be the main KPI in this research. The attractiveness based model combines container selection and route optimization to determines the best moment to collect a container, with the goal of minimizing the total travel time for the entire horizon. It determines on which day a container is the most attractive, i.e. when it costs the least amount of time. This enables the model to collect a container one day earlier than normal, if it is on the route of the previous day. The model calculates all routes for all days during the horizon, but only executes the routes of the first day. After collection the fill levels of the collected containers are set to zero and the true deposit rates are added to all fill levels. These new fill levels are the input for the next day. Again the containers are collected by 2 trucks with a maximum of 2 routes per truck per day.

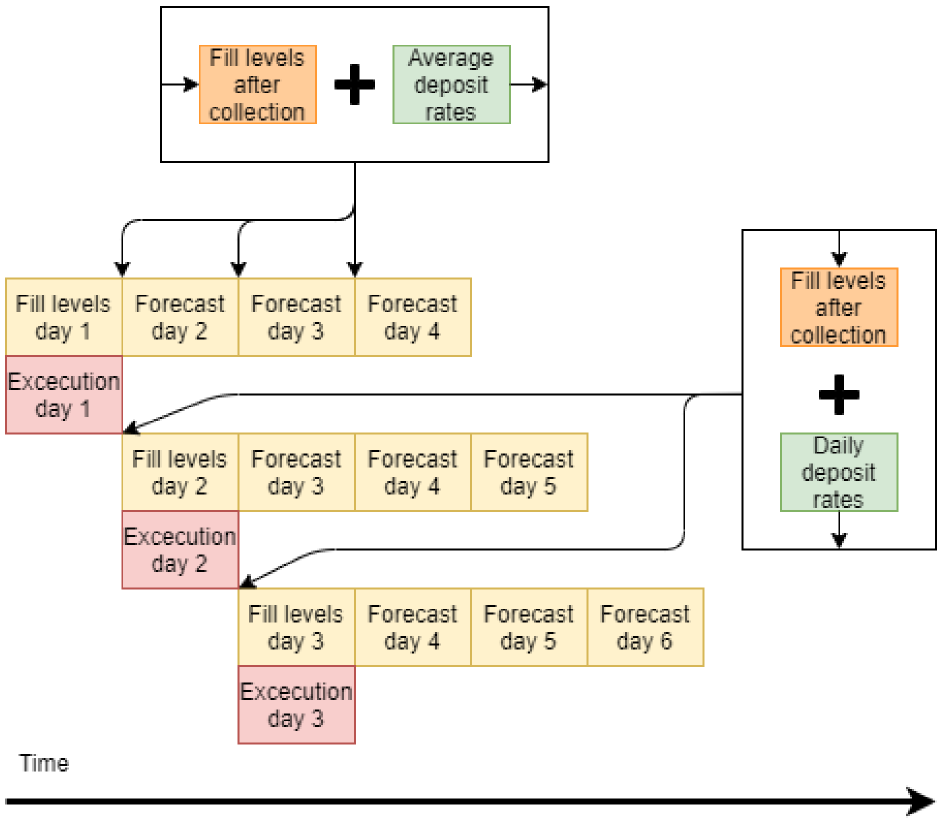

Figure 7: Rolling horizon approach for attractiveness and must-go may-go method

3.3.1 Differences from literature

The points about the use of real data and data cleaning were already made in the previous section. The only difference is the container selection method. Most articles use cost objective functions, which from a financial stand point makes sense. However, since AMCS and OMRIN use total travel time as the main KPI, this will also be used in this research. Since computational time is often an issue when large problems like the SWCP are solved mathematically, this research tries to minimize computational time by excluding containers with low fill levels. Over complicating a model also leads to high computational times and sometimes even to unwanted results. Therefore a simplistic forecasting model is used, in contrast to Markov et al. (2016). Ramos et al. (2018)

included a profit, obtained from the collected amount of waste, in their objective function. This was neglected in this research as containers will always be emptied rather later than sooner. This again simplifies the objective function and the model as a whole.

3.3.2 Mathematical model

Indices and sets:

i :	index for each node in the network.	$\forall i \in ned, nid \; or \; early$
j :	index for each node in the network	$\forall i \in ned, nid \; or \; early$
k :	index for the number of routes.	$k \in K$
day :	index for the number of days in the horizon.	$\forall day \in days$
ned :	set of nodes excluding depot and terminal	$[0, ..., 406]$
nid :	set of nodes including depot(408) and terminal (407, 409)	$[0, ..., 409]$
$early$:	set of nodes that need to be collected before 9:00 AM	$[23, 57, 71, 72, ...]$
K :	set for the number of routes needed	$[0, ..., total \; volume/Q]$
$days$:	set for the days in the horizon	$[0, 1, 2, 3]$

Decision variables:

$x_{i,j,k}$:	Binary variable which is 1 if arc i->j is used by vehicle k and 0 otherwise.
$se_{i,k}$:	Integer variable which is 1 for the first node, 2 for the second etc.
$w_{i,k}$:	Continues variable which equals the collected volume of truck k at node i.
$u_{i,k}$:	Integer variable which equals the time a node is visited, lower bound = 24300. Starting time = 24300 sec => 6:45 AM.
$hfl_{i,day}$:	Continues variable which equals the fill level of container i on a day in the horizon

Parameters:

$time\_matrix_{i,j}$:	Travel time between node i and node j, in seconds.
$Threshold\_level_i$:	Threshold levels for node i, in percentages.
$Container\_volumes_i$:	Total volume of the container for node i, in liters.
$Fill\_level\_list_i$:	Fill level of node i, in percentages.
st_i:	Service time of node i, in seconds.
Q :	Maximum load of each vehicle.
T :	Time before all containers in early need to be collected.

Objective function:

$$\text{MIN} \sum_{i \in nid} \sum_{j \in nid} \sum_{k \in K} \sum_{day \in days} x_{i,j,k,day} * time\_matrix_{i,j} + st_i * x_{i,j,k,day}$$

Constraints:

In ascending order, constraint (1) copies the fill levels of the total fill level list to the first day of the horizon. Constraints (2-4) fill the rest of the fill levels within the horizon with the influence of the collection of a container. Constraint (5) prevents overflows, constraint (6) reduces computational time by excluding containers with low fill levels. Constraint (7) will prevent arcs from and to the same node. Constraints (8-12) forces each route to start at node 407, visits node 408 if waste was collected and end at node 409. Note again that node 407 and 409 are two nodes representing the same terminal and node 408 is the deposit site. Constraint (13) ensures flow conservation at each node. Constraints (14-18) prevent sub tours by numbering of nodes and only allowing the visit of nodes with higher numbers. Constraint (19) sets the load at start point equal to 0. Constraints (20 and 21) ensure load continuity and constraint (22) limits the load of each vehicle. Constrains (23 and 24) ensure visit time continuity and finally constraint (25) makes sure all nodes in early are collected before 9.00AM. Note that in constraints (20), (21), (23) and (24) the big M method is used. The M displayed in the constraint represents the number 100000.

$$hfl_{i,0} = total\_fill\_level\_list_{i,0} \qquad \forall i \in nid \tag{1}$$

$$hfl_{i,1} = hfl_{i,0} * (1 - \sum_{i \in nid} \sum_{k \in K} x_{i,j,k,0}) + add_i \qquad \forall i \in nid, \tag{2}$$

$$hfl_{i,2} = hfl_{i,1} * (1 - \sum_{i \in nid} \sum_{k \in K} x_{i,j,k,1}) + add_i \qquad \forall i \in nid, \tag{3}$$

$$hfl_{i,2} = hfl_{i,1} * (1 - \sum_{i \in nid} \sum_{k \in K} x_{i,j,k,1}) + add_i \qquad \forall i \in nid, \tag{4}$$

$$hfl_{i,1} \leq 100 \qquad \forall i \in nid, \forall day \in days \backslash 0 \tag{5}$$

$$hfl_{i,day} \geq 40 * \sum_{j \in nid} \sum_{k \in K} x_{i,j,k,day} \qquad \forall i \in nid, \forall day \in days \tag{6}$$

$$x_{i,i,k,day} = 0 \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{7}$$

$$\sum_{j \in nid} x_{terminal,j,k,day} = 1 \qquad \forall k \in K, \forall day \in days \tag{8}$$

$$\sum_{i \in nid} x_{i,terminal,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{9}$$

$$\sum_{i \in nid} x_{i,terminal,k,day} = 1 \qquad \forall k \in K, \forall day \in days \tag{10}$$

$$\sum_{j \in nid} x_{terminal,j,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{11}$$

$$\sum_{i \in nid} w_{i,k} * (1 - x_{depot,terminal,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{12}$$

$$\sum_{j \in nid} x_{i,j,k,day} - \sum_{j \in nid} x_{j,i,k,day} = 0 \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{13}$$

$$se_{terminal,k,day} = 1 \qquad \forall k \in K, \forall day \in days \tag{14}$$

$$se_{i,k,day} \geq 2 \qquad \forall i \in nid \backslash \text{terminal}, \forall k \in K, \forall day \in days \tag{15}$$

$$se_{i,k,day} \leq |nid| \qquad \forall i \in nid \backslash \text{terminal}, \forall k \in K, \forall day \in day \tag{16}$$

$$se_{i,k,day} - se_{j,k,day} + |nid| * x_{i,j,k,day} \leq (|nid| - 1) \qquad \forall i \in nid \backslash \text{|nid|-1}, \forall j \in nid \backslash \text{terminal} \; \forall k \in K, \forall day \in days \tag{17}$$

$$se_{i,k,day} - se_{j,k,day} + |nid| * (1 - x_{i,j,k,day}) \geq -1 \qquad \forall i \in nid \backslash \text{|nid|-1}, \forall j \in nid \backslash \text{terminal} \; \forall k \in K, \forall day \in days \tag{18}$$

$$w_{terminal,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{19}$$

$$w_{j,k,day} \geq w_{i,k,day} + (hfl_j/100) * cv_j - M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \; \forall day \in days \tag{20}$$

$$w_{j,k,day} \leq w_{i,k} + (hfl_j/100) * cv_j + M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \; \forall day \in days \tag{21}$$

$$w_{i,k,day} \leq Q \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{22}$$

$$u_{j,k,day} \geq u_{i,k,day} + st_i + time\_matrix_{i,j} - M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{23}$$

$$u_{j,k,day} \leq u_{i,k,day} + st_i + time\_matrix_{i,j} + M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{24}$$

$$u_{i,k,day} = T \qquad \forall i \in early, \forall k \in K \tag{25}$$

3.3.3 Model optimization

The concept of the attractiveness model is to set a minimum amount of restrictions on the model and let the computer figure out what the best moment is to collect a container. In order to select the best moment for collection, a forecasting model is included in this model. This forecasting model will be used on every day. When a three day horizon is used, the fill levels for the next three days are examined and routes are suggested for all of them. After the optimization of this horizon the first day is executed and the horizon is moved up to the next day. The term rolling horizon is therefore used to describe a horizon that moves along in time. The number of days within the rolling horizon can be set from one, up to three days. This research will investigate the influence of the length of the rolling horizon on the attractiveness and the must-go may-go model and investigate if one of the models benefits more from a rolling horizon than the other. Since an attractiveness model like this is hard to solve, some restrictions on the model are necessary. Since containers need to be collected at some point and large overflows should be prevented. Therefore, an upper limit on the volume of a 100% is set for all containers. In some specific cases this upper limit will be set above 100%. Although this will cause more overflows, it will also give the model more possibilities to optimize. To minimize computational time we exclude certain containers from being considered of being collected. The constraint is set to exclude, for example, all containers that have fill levels below 60%. Note that in this research the exclusion level will not go above 60% since it will be in conflict with the set threshold levels. However, due to time limitations the exclusion level was not investigated. It showed to reduce computational times and therefore it was set to 60% for all runs.

To conclude, the attractiveness model has three tuning parameters:

- **Rolling horizon length**
- **Upper limit**
- **(Exclusion level)**

3.3.4 Verification

In this subsections the verification of the attractiveness model will be discussed and visualised. Each model is verified with the 10 tests listed below. Each test was performed using the same data set, and only critical parameters where added, removed or adjusted.

Continuity test: runs with slightly different parameters.
1) Basic run, with all constraints to test if the result is logic (bigger than 0).

Event tracing: check event order, causal relations, event times
2) Does the total time equals the service times and travel times.
3) Checking if order of visited nodes is in order and is equal to total visited nodes.

Degeneracy test: extreme cases.
4) Test without vehicle capacity constraint (only one vehicle should be used).
5) Test with vehicles capacity at 0.
6) Test with fill levels = 0 (no container should be collected)
7) Test with fill levels at 110 (all should be collected.
8) A test run where we remove the threshold level constraints, no container should be collected.

Consistency checks; e.g., doubled capacity à halved utilization.
9) Half the vehicles, double their capacity.

Fault injection: check whether faulty input is detected by the model.
10) Run without vehicles.

The table below contains the results of the verification tests. It describes the tested condition, the expected result, the obtained result and if the verification test was passed. This test was ran, using the AMCS data of 151 containers on 22=06-2020, obtained by the attractiveness model with a 1 day horizon and an absolute threshold level of 100%. These specifics were used due to large computational times when all 408 nodes were used and the fact that the 9th day of the aforementioned model used multiple vehicles to collect a reasonable amount of containers. From left to right it shows what is tested, what the expected result/behaviour is and what the actually measured result was. If the expected result meets the measured result, the test is passed.

Table 5: Verification results attractiveness model

Test	Expected results	Obtained result	Pass/Fail
1) Normal conditions/full model	Total time >0 Total volume > 0 Nr. of containers > 0 K >0	Total time = 15827 s Total volume = 128426 L Nr. of containers = 27 K = 2	Pass
2) Check event order	se[terminal, k, day] = 1 se[j, k, day] - se[i, k, day] = 1	se[terminal, k, day] = 1 se[j, k, day] - se[i, k, day] = 1	Pass
3) Total time check	Travel + service time = 15827 s	Travel + service time = 15827 s	Pass
4) No vehicle capacity limit	K = 1 Nr. of containers = +/- 27	K = 1 Nr. of containers = 28	Pass
5) Vehicle capacity at 0	Infeasible	Infeasible	Pass
6) All fill levels are 0	Total time = 0 K = 0	Total time = 0 K = 0	Pass
7) All containers are in early	All collected before 9:00AM Or infeasible	All collected before 9:00AM	Pass
8) No threshold level constraint	Total time = 0	Total time = 0	Pass
9) Double truck capacity	K = 1	K = 1	Pass
10) Run without vehicles	Infeasible	Infeasible	Pass

3.3.5 Validation

The validation of a model is the step where the researcher tests to what extend the computerized model represents the real system. Section 3.1 describes all the simplifications and details applied in the models of this research. As already mentioned, a full case study and comparison to AMCS was not possible. This limits the option to compare the model to reality and that way validate the models discussed in this research. However, all total driving times of AMCS for the test period are known, as well as the number of nodes visited. These can be used to see if the models presented in this research, provide solutions in the same order of magnitude as the solutions of AMCS. AMCS visited in the 12 days of the test period, 1006 nodes in a total travel time of 569580 seconds. However, the AMCS network consists of 408 containers and the one used in this research 151. The table below shows the comparison between the AMCS solution and the solutions of the attractiveness model.

Table 6: Validation results attractiveness model

	Total travel time [s]	Nr. of nodes visited
AMCS	569580	1006
AMCS Scaled	210800	416
Attractiveness (1 day horizon)	133129	288
Attractiveness (2 day horizon)	133083	302
Attractiveness (3 day horizon)	123868	294

These results show first of all that the results of this research are in the same order of magnitude as the results of a model which is successfully applied in real life. However, AMCS shows to have a higher (scaled) total travel time and a higher (scaled) number of visited nodes. Both of which are not necessarily positive. Visiting more nodes leads to higher total travel times, which for both AMCS and this research is the KPI that should be minimized. It is, however, important to note that AMCS also schedules custom orders and has to deal with more regulations than the models of this research and visits the city of Harlingen. As already mentioned, a comparison with AMCS at this point is very hard and therefore not very useful. These results only show that the results are of the right order of magnitude.

3.4 Must Go May Go model

The must-go may-go model combines the tho previous ones. As the name suggest containers can be collected based on one of two decisions. The first being its fill level passing the threshold level, this is called a must-go and it is threshold level based. The second is the collection of a container that has not yet passed its threshold level, but it costs less time to collect it today instead of tomorrow. This is called a may-go container and it is based on attractiveness. The idea it that the must-go may-go model has the best of both, previously mentioned, models. Due to the threshold levels the computational time should be reduced, while the attractiveness part allows for better optimization.

3.4.1 Differences from literature

The only article to compare with is the one of Mes et al. (2014). Mes et al. (2014) explicitly divide containers, each day, in one of three categories: must-go, may-go, or no-go. The algorithm uses a forecasting model to calculates the number of days till collection for each container. The models in this research do a similar thing, only constraints are used to determine which containers are must-gos, may-gos and no-gos. A deliberate decision was made to change the way attractiveness is defined, to differ from Mes et al. (2014). As Mes et al. (2014) compare the insertion cost of a container with its historically smoothed average, they calculate if the container is more attractive today then on average. But this is not the question. If a container has to be collected within the horizon, the question is when it is most beneficial to collect it. This should be completely independent of historical values.

3.4.2 Mathematical model

Indices and sets:

$i:$	index for each node in the network.	$\forall i \in ned, nid\ or\ early$
$j:$	index for each node in the network	$\forall i \in ned, nid\ or\ early$
$k:$	index for the number of routes.	$k \in K$
$day:$	index for the number of days in the horizon.	$\forall day \in days$
$ned:$	set of nodes excluding depot and terminal	$[0, ..., 406]$
$nid:$	set of nodes including depot(408) and terminal (407 and 409)	$[0, ..., 409]$
$early:$	set of nodes that need to be collected before 9:00 AM	$[23, 57, 71, 72, ...]$
$K:$	set for the number of routes needed	$[0, ..., totalvolume/Q]$
$days:$	set for the days in the horizon	$[0, 1, 2, 3]$

Decision variables:

$x_{i,j,k}$:	Binary variable which is 1 if arc i->j is used by vehicle k and 0 otherwise.
$se_{i,k}$:	Integer variable which is 1 for the first node, 2 for the second etc.
$w_{i,k}$:	Continues variable which equals the collected volume of truck k at node i.
$u_{i,k}$:	Integer variable which equals the time a node is visited, lower bound = 24300. Starting time = 24300 sec => 6:45 AM.
hfl_i:	Continues variable which equals the fill level of container i troughout the horizon.

Parameters:

$time\_matrix_{i,j}$:	Travel time between node i and node j, in seconds.
$Threshold\_level_i$:	Threshold levels for node i, in percentages.
$Container\_volumes_i$:	Total volume of the container for node i, in liters.
$Fill\_level\_list_i$:	Fill level of node i, in percentages.
st_i:	Sevice time of node i, in seconds.
Q :	Maximum load of each vehicle.
T :	Time before all containers in early need to be collected.

Objective function:

$$\text{MIN} \sum_{i \in nid} \sum_{j \in nid} \sum_{k \in K} \sum_{day \in days} \mathrm{x}_{i,j,k,day} * \mathrm{time\_matrix}_{i,j} + \mathrm{st}_i * \mathrm{x}_{i,j,k,day}$$

Constraints:

In ascending order, constraint (1) copies the fill levels of the total fill level list to the first day of the horizon. Constraints (2-4) fill the rest of the fill levels within the horizon with the influence of the collection of a container. Constraint (5) prevents overflows, constraint (6) reduces computational time by excluding containers with low fill levels. Constraint (7) forces containers to be collected if their fill level exceeds their threshold level. Constraint (8) will prevent arcs from and to the same node. Constraints (9-13) forces each route to start at node 407, visits node 408 if waste was collected and end at node 409. Note again that node 407 and 409 are two nodes representing the same terminal and node 408 is the deposit site. Constraint (14) ensures flow conservation at each node. Constraints (15-19) prevent sub tours by numbering of nodes and only allowing the visit of nodes with higher numbers. Constraint (20) sets the load at start point equal to 0. Constraints (21 and 22) ensure load continuity and constraint (23) limits the load of each vehicle. Constrains (24 and 25) ensure visit time continuity and finally constraint (26) makes sure all nodes in early are collected before 9.00AM. Note that in constraints (21), (22), (24) and (25) the big M method is used. The M displayed in the constraint represents the number 100000.

$$hfl_{i,0}= total\_fill\_level\_list_{i,0} \qquad \forall i \in nid \tag{1}$$

$$hfl_{i,1}= hfl_{i,0}*(1 - \sum_{i\in nid}\sum_{k\in K} x_{i,j,k,0}) + add_i \qquad \forall i \in nid, \tag{2}$$

$$hfl_{i,2}= hfl_{i,1}*(1 - \sum_{i\in nid}\sum_{k\in K} x_{i,j,k,1}) + add_i \qquad \forall i \in nid, \tag{3}$$

$$hfl_{i,2}= hfl_{i,1}*(1 - \sum_{i\in nid}\sum_{k\in K} x_{i,j,k,1}) + add_i \qquad \forall i \in nid, \tag{4}$$

$$hfl_{i,1}\ leq100 \qquad \forall i \in nid, \forall day \in days\backslash 0 \tag{5}$$

$$hfl_{i,day}\geq 40 * \sum_{j\in nid}\sum_{k\in K} x_{i,j,k,day} \qquad \forall i \in nid, \forall day \in days \tag{6}$$

$$hfl_{i,0}\leq threshold\_levels_i + 100 * \sum_{j\in nid}\sum_{k\in K} x_{i,j,k,0} \qquad \forall i \in ned \tag{7}$$

$$x_{i,i,k,day}= 0 \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{8}$$

$$\sum_{j\in nid} x_{terminal,j,k,day}= 1 \qquad \forall k \in K, \forall day \in days \tag{9}$$

$$\sum_{i\in nid} x_{i,terminal,k,day}= 0 \qquad \forall k \in K, \forall day \in days \tag{10}$$

$$\sum_{i\in nid} x_{i,terminal,k,day}= 1 \qquad \forall k \in K, \forall day \in days \tag{11}$$

$$\sum_{j\in nid} x_{terminal,j,k,day}= 0 \qquad \forall k \in K, \forall day \in days \tag{12}$$

$$\sum_{i\in nid} w_{i,k}*(1 - x_{depot,terminal,k,day}= 0 \qquad \forall k \in K, \forall day \in days \tag{13}$$

$$\sum_{j\in nid} x_{i,j,k,day} - \sum_{j\in nid} x_{j,i,k,day}= 0 \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{14}$$

$$se_{terminal,k,day}= 1 \qquad \forall k \in K, \forall day \in days \tag{15}$$

$$se_{i,k,day}\geq 2 \qquad \forall i \in nid\backslash \text{terminal}, \forall k \in K, \forall day \in days \tag{16}$$

$$se_{i,k,day}\leq |nid| \qquad \forall i \in nid\backslash \text{terminal}, \forall k \in K, \forall day \in day \tag{17}$$

$$se_{i,k,day} - se_{j,k,day} + |nid| * x_{i,j,k,day}\leq (|nid| - 1) \qquad \forall i \in nid\backslash \text{|nid|-1}, \forall j \in nid\backslash \text{terminal} \ \forall k \in K, \forall day \in days \tag{18}$$

$$se_{i,k,day} - se_{j,k,day} + |nid| * (1 - x_{i,j,k,day}) \geq -1 \qquad \forall i \in nid\backslash \text{|nid|-1}, \forall j \in nid\backslash \text{terminal} \ \forall k \in K, \forall day \in days \tag{19}$$

$$w_{terminal,k,day}= 0 \qquad \forall k \in K, \forall day \in days \tag{20}$$

$$w_{j,k,day}\geq w_{i,k,day} + (hfl_j/100) * cv_j - M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \ \forall day \in days \tag{21}$$

$$w_{j,k,day} \leq w_{i,k,day} + (hfl_j/100) * cv_j + M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \ \forall day \in days \tag{22}$$

$$w_{i,k,day} <= Q \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{23}$$

$$u_{j,k,day} \geq u_{i,k,day} + st_i + time\_matrix_{i,j} - M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{24}$$

$$u_{j,k,day} \leq u_{i,k,day} + st_i + time\_matrix_{i,j} + M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{25}$$

$$u_{i,k,day} = T \qquad \forall i \in early, \forall k \in K \tag{26}$$

3.4.3 Model optimization

Since the must-go may-go model is a combination of the previous two discussed models, also the tuning parameters are similar. Since the upper limit threshold level is redundant, it is not necessarily used as a tuning parameter. However, if it is left unchanged it will limit the system even if the threshold buffer is changed. The must-go may-go model results will therefore be characterised by four tuning parameters:

- **Threshold buffer**
- **Rolling horizon length**
- **(Exclusion level)**
- **Upper limit**

3.4.4 Verification

In this subsections the verification of the threshold model will be discussed and visualised. Further on in this report also the two other models will be verified. Each model is verified with the 10 tests listed below. Each test was performed using the same data set, and only critical parameters where added, removed or adjusted.

Continuity test: runs with slightly different parameters.
1) Basic run, with all constraints to test if the result is logic (bigger than 0).

Event tracing: check event order, causal relations, event times
2) Does the total time equals the service times and travel times.
3) Checking if order of visited nodes is in order and is equal to total visited nodes.

Degeneracy test: extreme cases.
4) Test without vehicle capacity constraint (only one vehicle should be used).
5) Test with vehicles capacity at 0.
6) Test with fill levels = 0 (no container should be collected)
7) Test with fill levels at 110 (all should be collected.
8) A test run where we remove the threshold level constraints, no container should be collected.

Consistency checks; e.g., doubled capacity à halved utilization.
9) Half the vehicles, double their capacity.

Fault injection: check whether faulty input is detected by the model.
10) Run without vehicles.

The table below contains the results of the verification tests. It describes the tested condition, the expected result, the obtained result and if the verification test was passed. This test was ran, using the AMCS data of 151 containers on 17=06-2020, obtained by the attractiveness model with a 1 day horizon and an absolute threshold level of 100%. These specifics were used due to large computational times when all 408 nodes were used and the fact that the 9th day of the aforementioned model used multiple vehicles to collect a reasonable amount of containers. From left to right it shows what is tested, what the expected result/behaviour is and what the actually measured result was. If the expected result meets the measured result, the test is passed.

Table 7: Verification results must-go may-go model

Test	Expected results	Obtained result	Pass/Fail
1) Normal conditions/full model	Total time >0 Total volume > 0 Nr. of containers > 0 K >0	Total time = 15858 Total volume = 132942 Nr. of containers = 30 K = 2	Pass
2) Check event order	se[terminal, k] = 1 se[j, k] - se[i, k] = 1	se[terminal, k] = 1 se[j, k] - se[i, k] = 1	Pass
3) Total time check	15858 = travel + service time	15858 = travel + service time	Pass
4) No vehicle capacity limit	K = 1 Nr. of containers = +/- 30	K = 1 Nr. of containers = 31	Pass
5) Vehicle capacity at 0	Infeasible	Infeasible	
6) All fill levels are 0	Total time = 0	Total time = 0	Pass
7) All containers in early	All collected before 9:00 AM Or infeasible	All collected before 9:00 AM	Pass
8) No threshold level constraints	Total time = 0	Total time = 0	Pass
9) Double truck capacity	K = 1	K = 1	Pass
10) Run without vehicles	Infeasible	Infeasible	Pass

3.4.5 Validation

The validation of a model is the step where the researcher tests to what extend the computerized model represents the real system. Section 3.1 describes all the simplifications and details applied in the models of this research. As already mentioned, a full case study and comparison to AMCS was not possible. This limits the option to compare the model to reality and that way validate the models discussed in this research. However, all total driving times of AMCS for the test period are known, as well as the number of nodes visited. These can be used to see if the models presented in this research, provide solutions in the same order of magnitude as the solutions of AMCS. AMCS visited in the 12 days of the test period, 1006 nodes in a total travel time of 569580 seconds. However, the AMCS network consists of 408 containers and the one used in this research 151. The table below shows the comparison between the AMCS solution and the solutions of the must-go may-go model.

Table 8: Validation results must-go may-go model

	Total travel time [s]	Nr. of nodes visited
AMCS	569580	1006
AMCS Scaled	210800	416
Must-go may-go (1 day horizon)	135115	294
Must-go may-go (2 day horizon)	147589	335
Must-go may-go (3 day horizon)	137379	319

These results show first of all that the results of this research are in the same order of magnitude as the results of a model which is successfully applied in real life. However, AMCS shows to have a higher (scaled) total travel time and a higher (scaled) number of visited nodes. Both of which are not necessarily positive. Visiting more nodes leads to higher total travel times, which for both AMCS and this research is the KPI that should be minimized. It is, however, important to note that AMCS also schedules custom orders and has to deal with more regulations than the models of this research and visits the city of Harlingen. As already mentioned, a comparison with AMCS at this point is very hard and therefore not very useful. These results only show that the results are of the right order of magnitude.

4 Experiments

In order to evaluate the three methods, numerical experiment were performed, based on real data obtain from AMCS. The details of this data will be discussed as well as the cleaning of it and the use of it in order to build the three models.

4.1 Data processing

The data used in this research originates from 408 general waste containers that are managed by Omrin. Two data sets were used, one for building the model, verification and for gaining insights. The other one was planed to be used for a test case to compare the three models of this research with the one of AMCS. This however was not possible due to very large computational times. The sets of data contain data recorded from 14-06-2020 to 13-07-2020 and respectively 20-10-2020 to 20-11-2020. This data consists of percentage fill levels of 408 containers, located in Leeuwarden and surroundings. Once per day the data is transferred in the form of an XML file, which contains information about each container, foremost its fill level. The term data will, from now on, refer to these fill levels and the daily deposit rates obtained from them.

4.1.1 Uncertainties and errors

As is the case with almost any data, it will contain some sort of errors. This section will explain which type of errors were observed in the data, which are accounted for and which are neglected. Some errors will not cause significant problems while others do. In reality, most of these errors cause problems and should be dealt with. For this research only certain errors need fixing, others can be neglected or removed entirely by removing that specific container from the network, if necessary. This is due to the fact that this research works with several computer models, instead of a real waste collection system.

An example of an error that will cause issues in reality but not necessarily for the models in this research if not dealt with quickly, is a zero reading. This is an error where the container sends wrong information, a fill level of 0%, and avoids the collection of this specific container. In case of a zero reading, it is more likely that the sensor or container is defect than that indeed no deposits have been made over a time span of multiple days. In such a case the container will never be selected for collection by any model. If the sensor is broken, the container could already be overflowing for many days when this error is not observed and fixed by the company. A similar thing occurs with readings of constant fill levels. One container showed a constant fill level of 26% throughout the entire data set. Since this research uses the daily deposit rates, containers like the one just mentioned will stay on their initial fill level and will never be collected. Another container showed a fill level of 139% for 14 out of the 31 days of the collected data and 0 for the last 15 days. Since its fill level is above any threshold level and above 100% it will be collected on the first day and be simulated as empty from that point on. Because this research works with the daily deposit rates and not directly with the fill levels obtained from the XML file these errors will not cause a major problem for the models discussed in this research. For every containers' data it holds, if the containers' fill level is higher than the threshold level, this container will be collected on the first day and remains uncollected for the rest of the test period. If its fill level is below the threshold level it will remain uncollected anyway. In reality, this container needs to be visited and fixed to prevent overflows and ensure correct collection. The data also shows cases where a container exceeds its threshold level but is not collected. Meaning the container overflows for multiple days in a row. A cause for this is when a container is not collectable, meaning the collection truck can not reach it. Streets can be closed and cars or construction work can block entrance to the container. If more information is obtained by the truck driver, it could be arranged that the container is removed from collection by a certain amount of time.

In this research, these containers do not cause any problems since the daily deposit rates can still be obtained and these models are not limited by physical obstacles. To be able to compare results with AMCS, a second data set (14-06-2020 to 13-07-2020) was cleaned and used. This data set showed constant fill levels for all containers over a period of 2 days. As it is unlikely that all 408 containers had a malfunction during these 2 days, for some reason old measurements were repeatedly sent to AMCS. Since fill levels show a highly linear relation, as can be seen in the graph below, inter- and extrapolation was used to fill up these wrong data points.

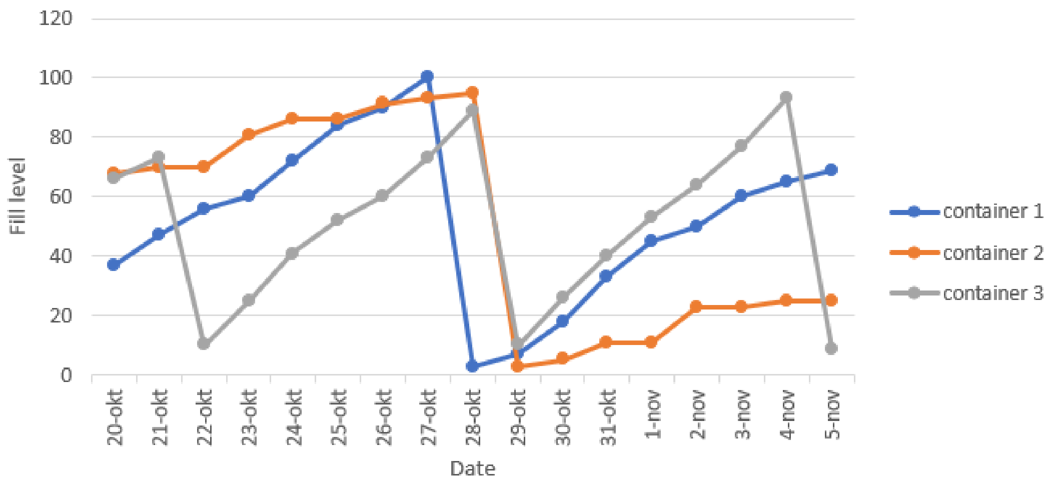

Figure 8: Linear relation fill levels

When the fill level of a container was lower before the two-day malfunction than after, linear interpolation was applied to these data points. This method uses the linear trend of the known data points and calculates the missing values accordingly. In this case, interpolation was far easier than extrapolation as the trends are not continuous, they range approximately from 0 to 100%. When the fill level of a container was higher before the two-day malfunction than after, this container had to be emptied in the mean time, but the day and the fill level at the time of collection are unknown. If possible, extrapolation was applied first, using the trend of the data before the malfunction. Data points were created using this method up to the point of collection, which is often close to 100%. If any data points were still open, extrapolation was used using the trend from data point after the malfunction. If no or to little data point were available before the malfunction, first the extrapolation was done using data points after the malfunction. Often the final missing data points could now be filled. Especially for fast running containers, i.e. a container with high daily deposit rates, not a lot of data points were available to use for inter- or extrapolation. In these cases, data point were created based on the rough trend-line multiple days ago, but still from the same container. As the first day of the malfunction was a Sunday, and on Sundays no collections are performed, the collection had to be made before the malfunction, on the second day of the malfunction or later.

Uncertainty always plays a role when measurements are performed. In the case of waste containers, managed by OMRIN, the uncertainty is caused by errors in the measurement but in the calculation. The number of valve movements is a measurement that can be done with almost zero uncertainty. What does have a contribution to the uncertainty, is the used volume or density of the average deposit. Since this is an average, the volume in most containers will differ from the calculated value, even if it is just a little bit. Another contribution to the uncertainty is caused by the fact that the measurement and the collection do not occur at the same time. As the time between measurement and collection increases, the chance that deposits are made increases as well and therefore the uncertainty of the containers' volume grows as well. Apparently, the average deposit volume used by OMRIN per valve movement works sufficiently , as they do not recall having route failures and have an acceptable amount of overflows. Note that route failures will occur when uncertainty is present but not taken account for. Safety margins often are the go-to solution. OMRIN also measures the weight during collection. By lifting the underground containers out of their foundations the total weight is measured. Subtracting the weight of a container and the total weight of the waste is determined. They are therefore able to check if the measured/calculated weight was indeed in the container. Since OMRIN uses a calculation to go from valve movements to weight/volume the are also able to influence the average uncertainty over all containers.

4.1.2 Cleaning sensor data

To be able to clean data it should be clear for what purpose the data will be used. For this research, the data consist of one XML file per day containing information about each container, the fill levels of all containers being the most important. Figure 9 shows the fill levels for three different containers, for 10 consecutive days. The models discussed in this research will be tested for a given time period. To start this period, all fill levels during the start-day should be known and will be directly copied from the XML file, into each model. After the start-day, the models will deviate not only from each other but foremost from the measured fill levels, measured by the sensors in each container, for the consecutive days. For all consecutive days, the daily deposit rates can be obtained by subtracting two consecutive fill levels for the same container. These daily deposit rates can be used for each model and are shown in Figure 10, for the same containers as Figure 9. When a model decides to collect a container its fill level will be set to 0 after which the daily deposit rate for that container during that day will be added. When a container is not emptied, the daily deposit rate will be added to the initial fill level of that container for that day. A problem arises when a container is emptied in real life. Now the new fill level will be lower than the previous and this results in a negative daily deposit rate, as shown in red in Figure 10. To solve this problem negative daily deposit rates are replaced by the average daily deposit rate, calculated for all other days where positive daily deposit rates were known.

Table 9: Fill levels

	14-jun	**15-jun**	**16-jun**	**17-jun**	**18-jun**	**19-jun**	**20-jun**	**21-jun**	**22-jun**	**23-jun**
135817	62	76	91	15	27	43	61	72	83	101
143773	56	60	64	69	5	14	19	21	25	30
158743	89	103	4	9	15	21	32	41	52	60

Table 10: Daily deposit rates

	14-jun	**15-jun**	**16-jun**	**17-jun**	**18-jun**	**19-jun**	**20-jun**	**21-jun**	**22-jun**	**23-jun**
135817	14	15	-76	12	16	18	11	11	18	16
143773	4	4	5	-64	9	5	2	4	5	6
158743	14	-99	5	6	6	11	9	11	8	3

At first, an area in Arnhem, where Sues is responsible for the collection of was proposed to use as a test case and to use its data in general. Due to the size of this area, 257 containers, it seemed to be ideal for my thesis. Unfortunately, analysis of the data of Sues showed otherwise. Many containers gave 0 readings, meaning the sensor does not measure anything in the container throughout the test period of 30 days. Other containers showed varying fill levels which also decreased from time to time. In general, this is due to one of three reasons: First, the collection of a container, which would show a significant drop in fill level, +/- 40% and up. Second, a collapse of the pile of waste, which should show a small drop of the fill level +/- 10% or less.

And third, a wrong reading of the sensor. As long as the first and second reasons are far apart in terms of fill level drops, they should be easily distinguishable. But in the range between 10 and 40% it is hard to say which is which. This can be checked by comparing these drops to the actual list of collected containers.
Another issue with the Sues data is that it shows no fill levels above 80%. This is apparently due to a mistake in the sensor settings. The depth of the container is not equal to the range of the sensor, explaining the gap of 20%.

A supposedly better test area would be parts of the province of Friesland. Here Omrin is responsible for the container collection and the areas AMCS is involved in are Harlingen, Heereveen and Leeuwarden. Harlingen is quite small and is limited by only having 2 collection days, Heereveen is not yet operational, leaving Leeuwarden, including its surroundings as the only option. Leeuwarden has 408 containers that need to be collected making it a larger area than Sues in Arnhem. Omrin works with a valve movement sensor instead of an ultrasonic volumetric sensor. Of course the valve movements have to be translated to fill percentages but apparently, Omrin is quite successfully capable of doing this, since they do not recall having route failures and the number of overflows is acceptable. OMRIN is using an average of 37,5 Liters per valve movement. This is derived from a 4,5 m^3 container, which can hold 120 garbage bags, which should equal to an amount of 360 kg of waste. On the first glance the data of Omrin shows sigificant improvement when zero reading containers and constant fill levels are considered. Also the fill levels above 80% are measured and recorded. As discussed above, the two-day malfunction needed to be fixed however.

Although AMCS receives 1 set of measurements per hour (24 times a day), the set of containers only communicate their fill level once a day. I came across this result when I wanted to check at what times most deposits were being done. The resulting table showed only one positive change in fill level per day and these were mostly between 10 PM and 3 AM. After discussing these strange results with my supervisor, he and other employees of AMCS remembered that the containers only communicate their fill level once a day.

To create a usable set of daily deposit rates two successive fill levels where subtracted and when the result was negative, meaning it was emptied, it was replaced by the average daily deposit rate of that container.

4.1.3 XML files

In this research, all data is acquired through AMCS group Rotterdam in collaboration with OMRIN. The data consists of XML files that contain fill levels and other general information of waste containers such as location, type and size. These containers are located in Friesland in the cities of Harlingen, Heereveen and Leeuwarden. The XML files of OMRIN contain over a thousand containers with many different attributes. Depending on the specific area and part of this thesis certain containers and certain attributes need to be read from the XML file and used. For the area of Leeuwarden, 408 containers out of 1043 need to be considered and used.
A python script was build to read and copy container ids, coordinates, waste types, waste volumes and fill levels from these XML files. After obtaining the ids, coordinates, waste types and waste volumes, only the fill levels need to be copied from the XML files. And as explained in the previous section, the fill levels were used to obtain the daily deposit rates.

4.1.4 Distance and time matrices

To be able to minimize travel distance, distances between all nodes (containers, disposal sites and terminals) need to be known. Using the Here routing API distances between two nodes can be calculated when coordinates are given. For the test case of Leeuwarden, 411 nodes are taken into account, 408 containers, 1 disposal sight and 2 nodes for the terminal. A functional way of storing and displaying this information is in a distance matrix. This matrix contains distances from n nodes to n nodes, giving a n x n matrix. Since the length of a route from node A to node B is not necessarily equal to the length of the route from node B to node A, the distance matrix is not entirely symmetric. Note that some specific parts of the distance matrix are symmetric. Another thing to notice, is that all diagonal elements are zero since the distance from node n to node n is always zero. The final result is a 411 x 411 matrix, containing 168.921 elements. The python script used to calculate the distance and time matrices first makes a list of all coordinates of all the 411 nodes. Now the Here routing API is included in a loop which first fills a row with all distances from node i to all nodes 1 to n. This row is added to a list and this continues until 411 rows are added to this list. The result is a 411 x 411 distance and time matrix. The data was validated by a small check to see if the results would be generally acceptable and by comparing the results of the Here API with results from Google maps. Tables 11 and 12 show that the results of the Here API are acceptable, both tables show zeros on the diagonals and values within an acceptable range, considering the size of Leeuwarden. When compared to the results of Google maps, 13 and 14, there are some differences. First of all, the results from Google maps are less accurate than those of the Here API. Other differences can be due to specific settings in both programs, as both calculations were not executed simultaneously.

Table 11: HERE API travel distances [m]

		TO			
		node 0	**node 1**	**node 407**	**node 408**
FROM	**node 0**	0	2532	5185	3635
	node 1	1970	0	5132	3571
	node 407	5157	5646	0	3577
	node 408	3594	4027	3592	0

Table 12: HERE API travel times [s]

		TO			
		node 0	**node 1**	**node 407**	**node 408**
FROM	**node 0**	0	387	592	481
	node 1	391	0	623	537
	node 407	583	631	0	454
	node 408	436	541	475	0

Table 13: Google maps travel distances [m]

		TO			
		node 0	**node 1**	**node 407**	**node 408**
FROM	**node 0**	0	2500	4800	3300
	node 1	2000	0	5700	2700
	node 407	5100	5600	0	3600
	node 408	3400	2400	3600	0

Table 14: Google maps travel times [s]

		TO			
		node 0	**node 1**	**node 407**	**node 408**
FROM	**node 0**	0	420	540	420
	node 1	420	0	540	420
	node 407	540	540	0	360
	node 408	480	360	360	0

4.1.5 Forecasting

Forecasting is an often used and very helpful tool. In the case of waste collection and in particular smart waste collection, we consider quantitative forecasting. Quantitative forecasting is used when historical data is known and it can be assumed that the trends of this historical data in some form will continue into the future. For smart waste collection and this research, the first condition is definitely satisfied as a lot of data is recorded. The second condition is satisfied as well since all fill levels follow the same basic pattern, they will be emptied at some point and from then on the container will fill up gradually. Of course, different containers will fill up at different speeds but according to the data, most containers follow roughly the same trend over and over. Although it sounds and seems like forecasting always improves the solution this is definitely not always the case. The example of Markov et al. (2016) shows that it can also lead to less favourable results. And to quote of Margaret Heffernan:

“Efficiency works really well when you can predict exactly what you’re going to need. But when the anomalous or unexpected comes along, kids, customers, coconuts, well than efficiency is no longer your friend”

In case of a perfect forecasting model, which exactly predicts the fill levels for the coming days, the optimal solution for the SWCP should be found. Of course, the idea of perfect forecasting is nice, but in reality, it is also almost impossible. In this research, simplistic forecasting will be used which consists of an average daily deposit rate. The forecasting is kept simplistic, firstly due to time constraints and secondly because a detailed forecasting model is not necessary to answer the main research question. However, since for this research all daily deposit rates are known upfront, it is possible to run the models with a forecasting model that is 100% correct. Instead of only forecasting with an average deposit rate, a true forecasting model will be used as well. This true forecasting model will only differ in the fact that the average deposit rates will be swapped for the actual daily deposit rates. This model therefore should set an upper limit, and show how much improvement could be made by improving the forecasting model. For now, an average of the daily deposit rate per container will be used as the forecasted daily deposit rate. Note, this is container depended. Since data is lost due to the collection of containers, i.e. showing as negative deposit rates, the average is calculated using only positive deposit rates.

4.2 Results

This section will show the results of the models, both individually and in comparison to each other. The section will start with general results that apply to all models. Then each model is optimized and finally, the models are compared to each other based on total travel time and number of overflows, as the KPIs for the SWCP.

4.2.1 Used devices

During this research two devices were used to obtain results. This was mainly done due to large computational times. This section will first show the details of the devices and later explain what the consequences are of the usage of multiple devices.

HP ZBook Studio G5 Laptop

Operating sytem:	Windows 10
Processor:	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 2208 MHz 6 core(s), 12 logische processors)
Installed RAM:	16 GB
Used software:	Python 3.8 and Gurobi 9.0.3

ASUS PC

Operating system:	Windows 8
Processor:	Intel(R) Core(TM) i7-4470H CPU @ 3.50GHz, 3.50 GHz
Installed RAM:	32 GB
Used software:	Python 3.8 and Gurobi 9.0.3

Although codes and inputs are completely identical, the use of multiple devices can cause different results for the same model. This phenomenon is called the *Performance Variability.* Gurobi makes many decisions and therefore the branch-and-bound tree path can vary, for example, due to difference in computer hardware. To show however that the difference is minimal, on both devices the same must-go may-go instance was calculated, with a one- and two-day horizon. Table 15 and 16 show that in some instances, the outcomes can be identical except for the computational time which is a bit faster for the PC. In the two-day horizon a slight difference in container selection can be observed. Here the HP Zbook selects three additional containers that influence the route slightly.

Table 15: Results device comparison must-go may-go 1 day horizon

	ASUS PC	**HP Zbook**
Total travel time [s]	6765	6765
Total volume [L]	36193	36193
Nodes visited[-]	47, 50, 150, 133, 118, 17, 94, 25, 56, 143, 110	47, 50, 150, 133, 118, 17, 94, 25, 56, 143, 110
Computational time [s]	221	276

Table 16: Results device comparison must-go may-go 2 day horizon

	ASUS PC	**HP Zbook**
Total travel time [s]	8343	8895
Total volume [L]	45335	51510
Node visited[-]	47, 50, 133, 150, 56, 94, 97, 118, 17, 90, 25, 143, 110, 79	47, 55, 60 50, 150, 118, 97, 90, 94, 7, 21, 133, 110, 25, 143, 56
Computational time [s]	816	583

4.2.2 Overflow limit

As OMRIN recorded 120 overflows in the time period from 14-06-2020 to 28-06-2020 and 86 overflows when the warm-up period was excluded. This will be used as a baseline for the acceptable amount of overflows. However, since this research uses a network of only 151 nodes instead of 408, the number of allowed overflows has to be scaled down as well. Multiplying the 86 overflows with the ratio of 151/408 nodes it shows that the acceptable number of overflows for a network of 151 nodes and a time span of 12 days is 32. All models will be optimized to a level where the number of overflows will be around 32 and the total time is minimized. To verify this number also the amount of overflows were counted for 15 days in the data set of 20-10-2020 to 03-11-2020. This data shows 124 overflows for 408 containers during a 15 day period and 96 for a 12 day period. The number of overflows are important since there is a relation between the number of overflows and the total travel time. As parameters are set in such a way that the model gets more 'freedom' the amount of overflows increase. However, this allows the model to make more, and therefore better decisions and obtain a lower total travel time. The next section will show the relation between the number of overflows, the total travel time and the changing parameters.

4.2.3 Warm up period

As can be seen in the Figures 9 to 14, all models show a low collected volume of waste for the first day. Some even for a second day and others a high amount on the second or third day to compensate for the previous. These values are caused by a change of models. The original data, i.e. the fill levels of the first day, is influenced by the model or system that was used during the collection up to that point. This is the model AMCS uses to select containers for collection. Since from that point, other models are used a so-called warm-up period is shown. This is a period in which the data is not yet typical for the new model and this can be seen in the results. After some time the influence of the previous model is no longer present. To be able to draw the right conclusions and show the true effect of these models a warm-up period of 3 days is removed from the data. This means that the first 3 days of results will not be taken into account for analyzing the results of any of the models.

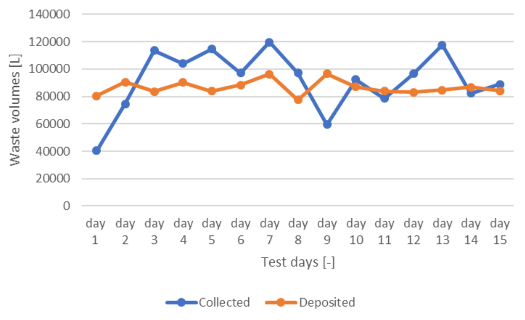
Figure 9: Waste volumes threshold 10% buffer

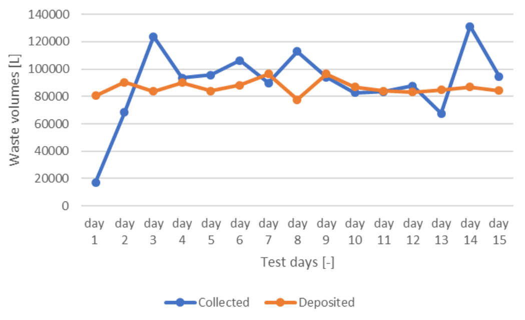
Figure 10: Waste volumes threshold 4% buffer

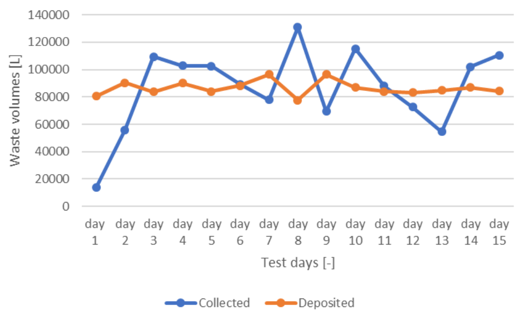
Figure 11: Waste volumes attractiveness one-day horizon

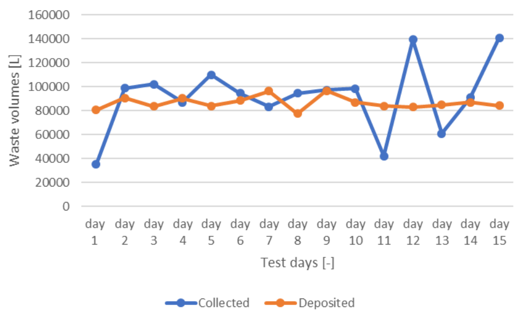
Figure 12: Waste volumes attractiveness two-day horizon

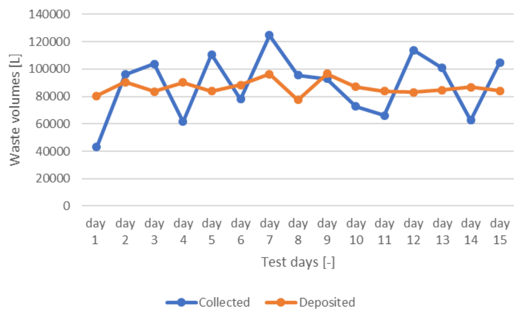
Figure 13: Waste volumes must-go may-go one-day horizon

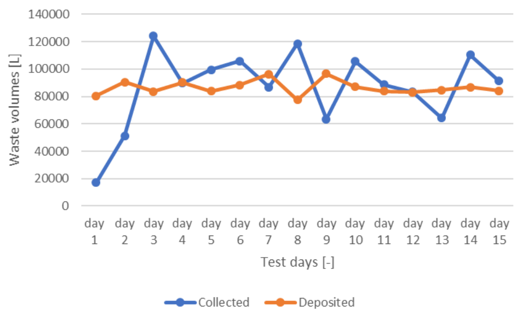
Figure 14: Waste volumes must-go may-go two-day horizon

4.2.4 Threshold based model

To optimize the threshold model, only one parameter can be changed that will influence the selection of containers. The threshold model instances were therefore calculated, using a network of 151 nodes and different values for the threshold buffer. One set of results of the threshold model is shown in Table 20. As can be seen, the first 3 days of the run are removed as mentioned above. All results are listed in Appendix B, in the same format as table 20. These results are combined in bar graphs, which will be used to represent the results of all three models.

Table 17: Results threshold model
Threshold buffer = 5%

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	11932	104031	2	8	23
Day 5	17718	114626	0	23	26
Day 6	11842	97176	0	11	27
Day 7	15084	119680	1	10	23
Day 8	13318	97088	2	7	25
Day 9	12209	59543	1	2	17
Day 10	10816	92238	2	6	20
Day 11	11460	78591	1	10	21
Day 12	12586	96501	1	7	26
Day 13	14468	117643	1	24	22
Day 14	12110	82292	0	8	23
Day 15	12811	88870	0	10	19
Sum	156354	1148279	11	126	272

Since for the threshold model, the fill levels can be calculated without finishing the entire route optimization, the results for many different buffer values could be calculated. Table 18 shows that a -1% buffer fits the baseline for overflows best and also shows a lower total travel time.

Table 18: Number of overflows for different buffer values

Amount of overflows [-]	69	38	32	24	11	5
Value for threshold buffer [%]	-5	-2	-1	0	5	10

Not all instances were fully calculated, but for the ones that were, the results are shown in Figure 15. This figure clearly shows a negative correlation between the number of overflows and the total travel time, i.e. when the number of overflows goes up, the total travel time goes down. Note that the threshold buffer step is not always equal to 5%.

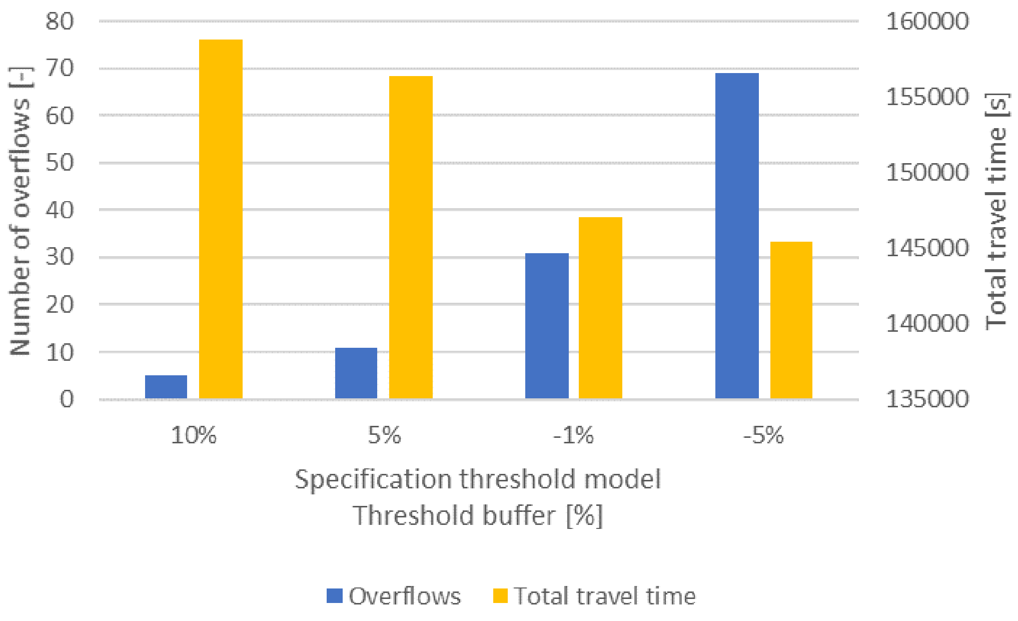

Figure 15: Results threshold model

4.2.5 Attractiveness based model

To optimize the attractiveness model, two parameters will be changed that will influence the selection of containers, namely the length of the horizon and the upper limit. As with the threshold model all calculated instances are joined in Figure 16. In contrast with Figure 15, here the total travel time does not show a clear negative correlation with the number of overflows. This is probably due to the fact that in this figure multiple variables are changed. It makes more sense for the attractiveness model and must-go may-go model to keep the horizons separate as well. Figure 17, 18 and 19 show the results per horizon.

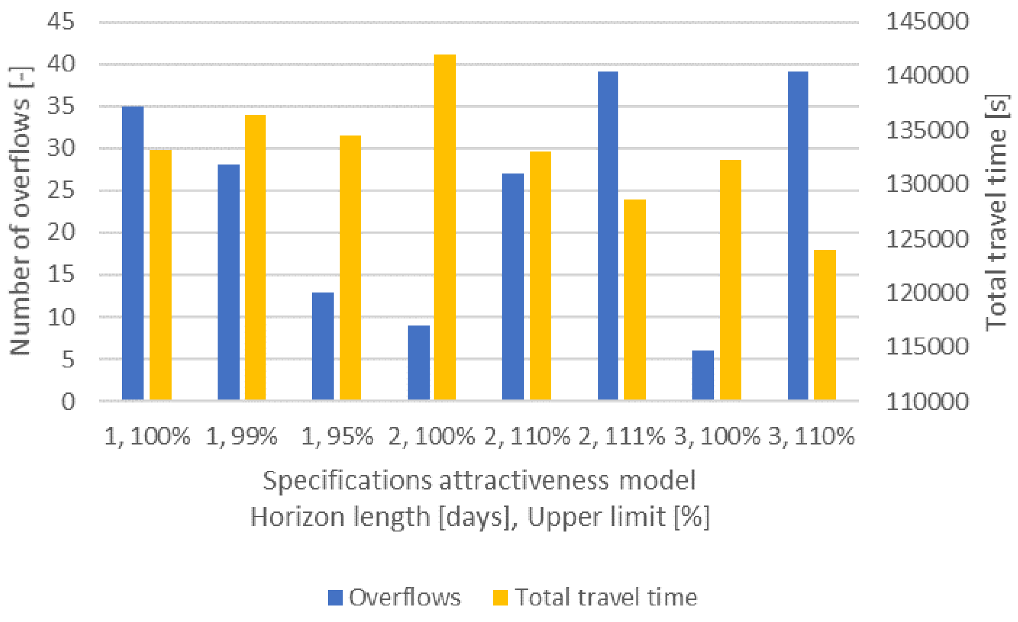

Figure 16: Results attractiveness model all instances

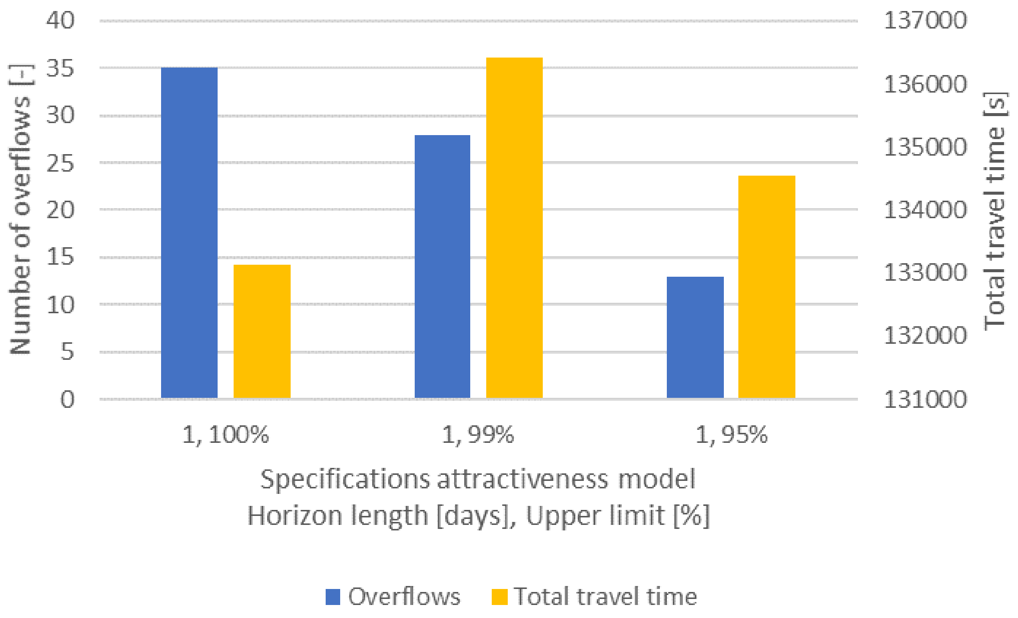

Figure 17: Results attractiveness model with a one-day horizon

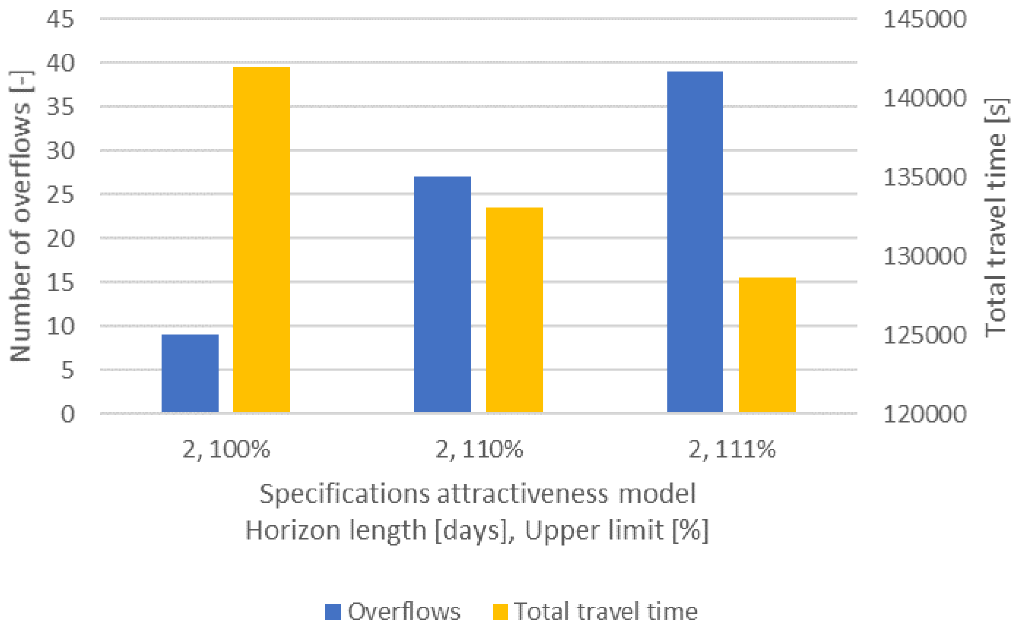

Figure 18: Results attractiveness model with a two-day horizon

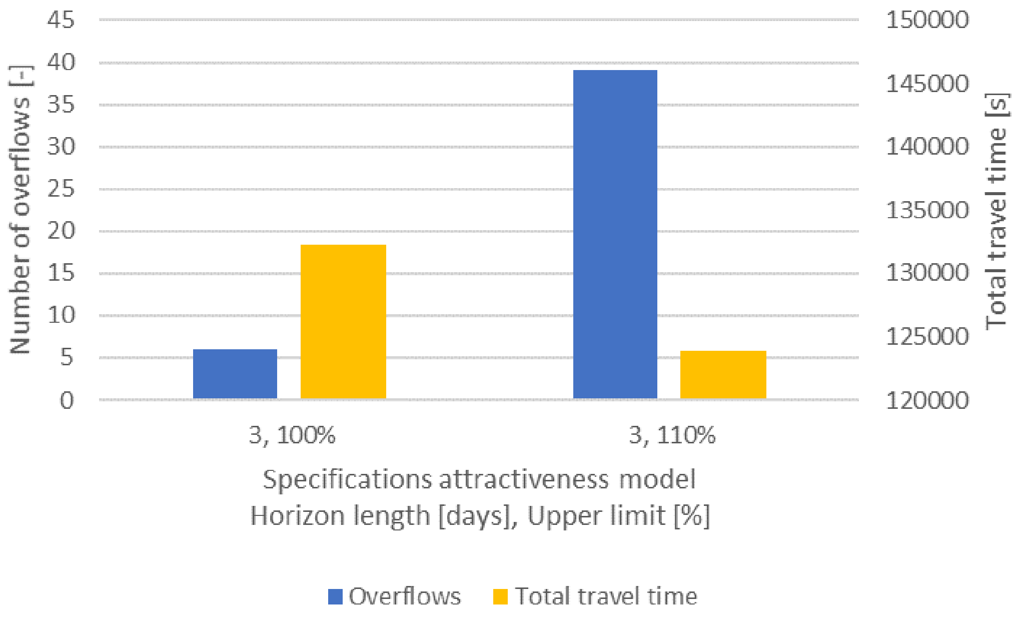

Figure 19: Results attractiveness model with a three-day horizon

The first calculated instances of each horizon length were run with an upper threshold limit is set to 100% and the exclusion level is set to 60% in order to reduce computational times. When the total travel time is observed, for the one-, two- and three-day horizon with an upper limit of 100%, it is odd to see that the model with the two-day horizon performs worse than both the model with the one-day horizon and the three-day horizon instance. This can be explained since the number of overflows is observed as well. The model with the one-day horizon already reached the baseline set for the number of overflows, while the other two models can possibly improve there total travel time by allowing more overflows.

For the one-day horizon instance the upper limit is decreased since the overflow baseline is already reached. For the two- and three-day horizon instances the upper limit is increased to allow more overflows for the models. The results show a significant improvement in the form of a 6% decrease in the total travel time, for both the two and three-day horizon attractiveness models. This came at the cost of an increase of overflows by 300% and 650% respectively. To conclude, the attractiveness model shows the lowest total travel time for the instance with a three-day horizon and an upper limit of 100%, while staying below the baseline of 32 overflows. All detailed results are displayed in Appendix B.

4.2.6 Must-go may-go based model

To optimize the must-go may-go model, three parameters can be changed that will influence the selection of containers. The overall results of the must-go may-go model are shown in Figure 20. The same as was true for the attractiveness model, also here no clear negative correlation between the number of overflows and the total travel time can be observed.

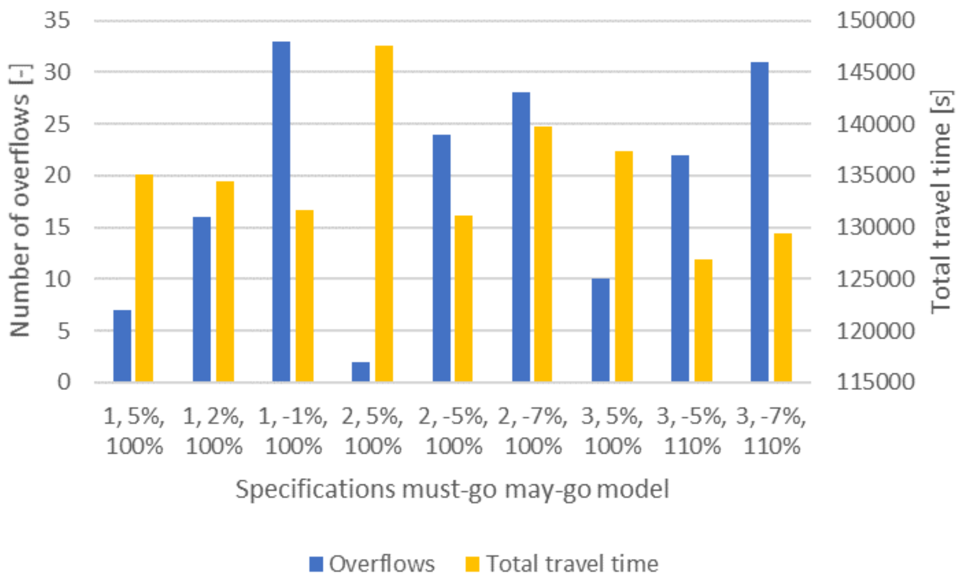

Figure 20: Results must-go may-go model

Note that while the one-day instances in 21 shows a clear negative correlation between the amount of overflows and the total travel time, the two- and three-day horizon instances in 22 and 23 show that increasing the number of overflows can also result in an increase of total travel time.

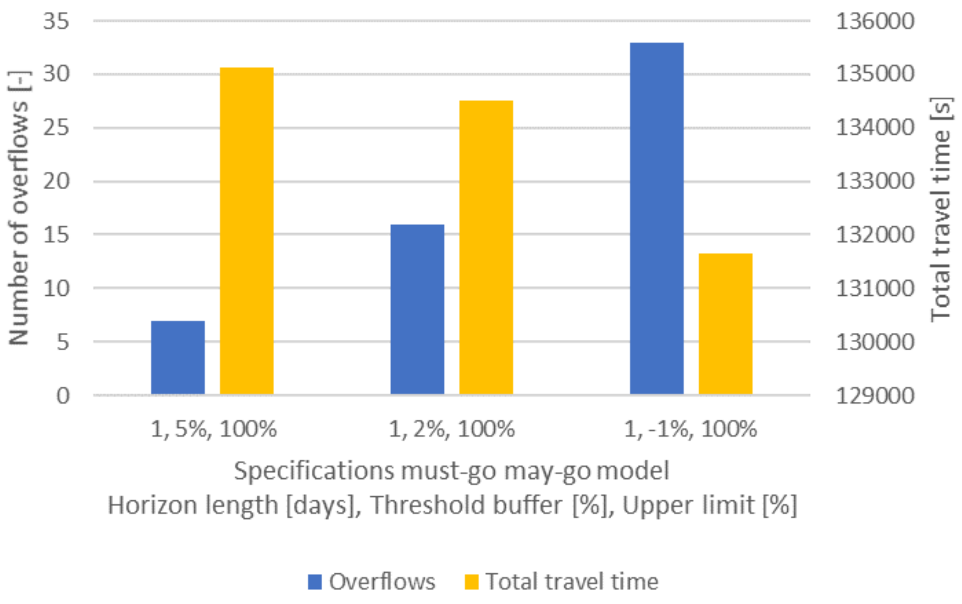

Figure 21: Results must-go may-go model with a one-day horizon

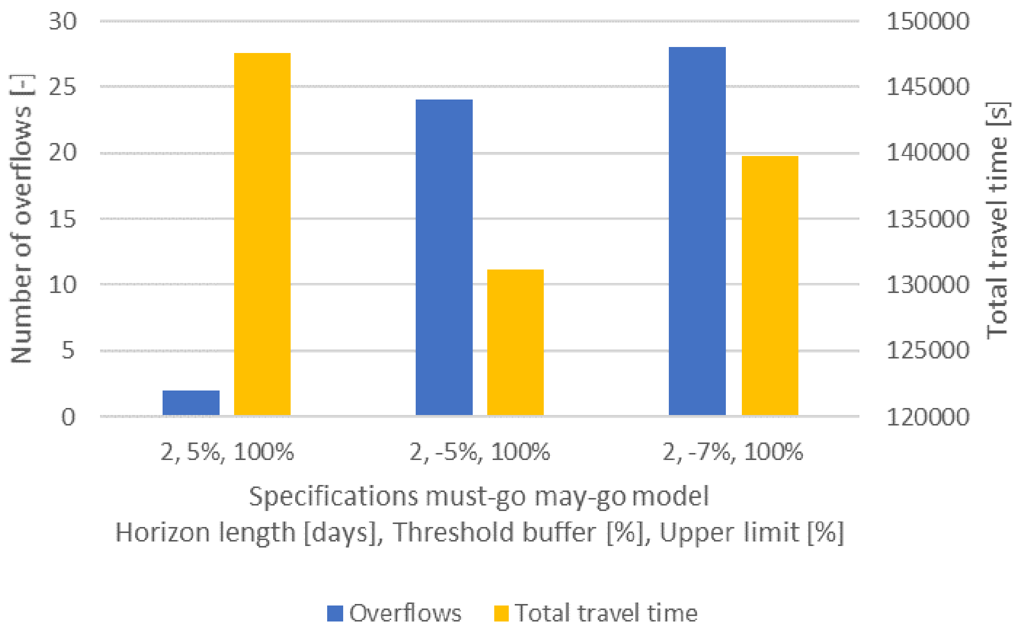

Figure 22: Results must-go may-go model with a two-day horizon

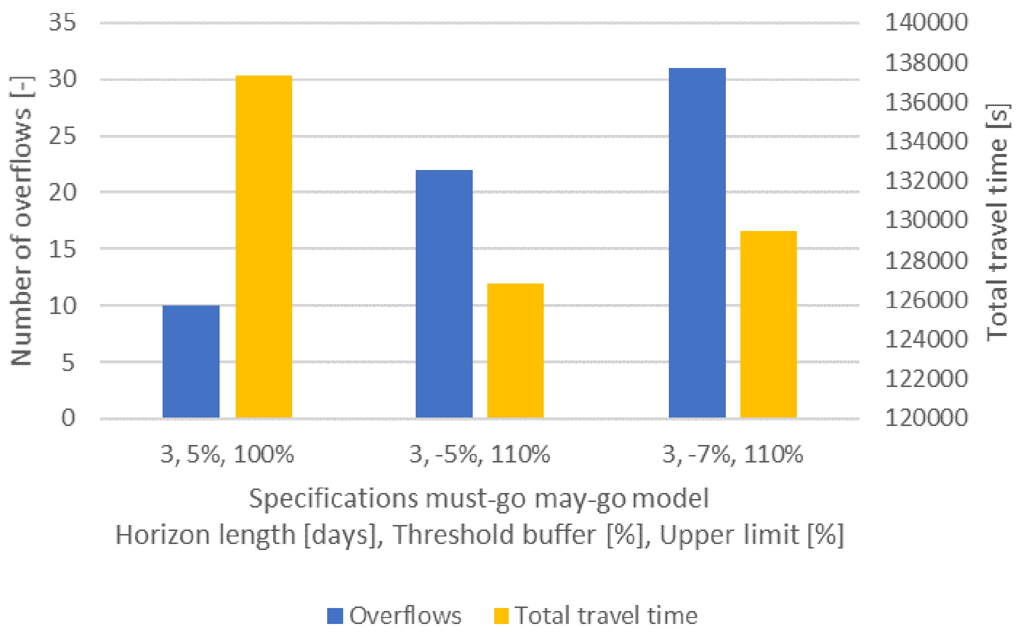

Figure 23: Results must-go may-go model with a three-day horizon

The three-day horizon instance shows the lowest total travel time when a threshold buffer of -5% is used and an upper limit of 110%. It stands out that the one-day horizon of the must-go may-go model reaches the overflow baseline already at a threshold buffer of -1%, while the two- and three-day horizon models do not even reach the overflow baseline at a threshold buffer of -7%. The negative threshold buffer means that the expected amount of waste for the next day exceeds the 100% and thus overflows are expected to occur more often.

4.2.7 Comparison of models

Figure 24 shows clearly that for now the must-go may-go model has the best performance considering the chosen KPIs. It also shows that more overflows are allowed for both the attractiveness and must-go may-go model and therefor other instances could still hold better solutions.

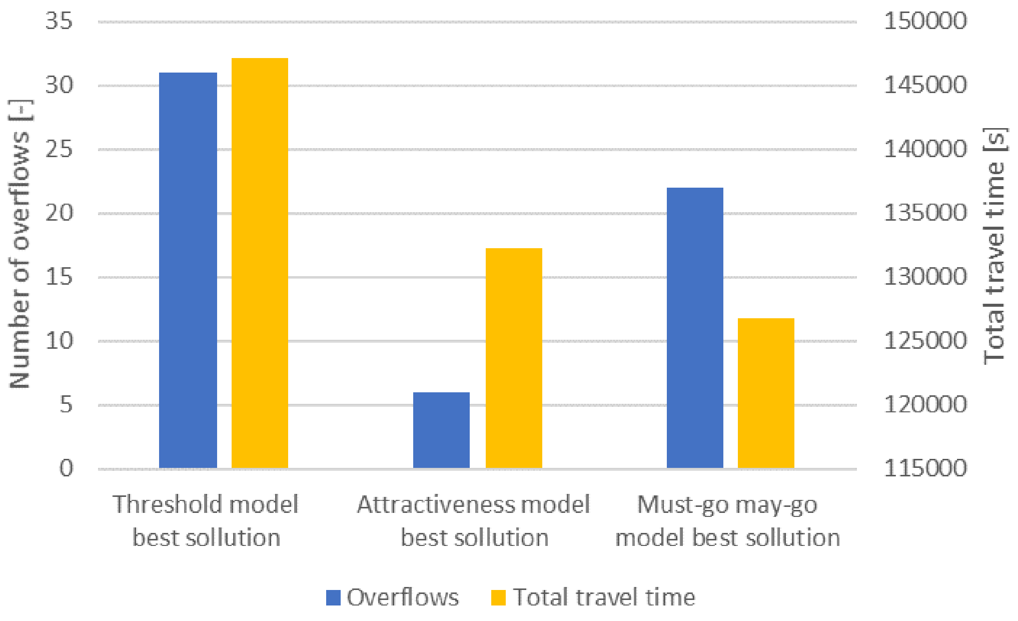

Figure 24: Results model comparison

In order to estimate how much improvement can be made using a very detailed forecasting model, a true forecasting model was used to compare to both the attractiveness model and the must-go may-go model. This true forecasting model only differs from the fact that the normal attractiveness and must-go may-go model use average daily deposit rates for their rolling horizon forecasts, while the true forecasting model uses the actual daily deposit rates. Therefore its forecasting is 100% accurate. The true forecasting model was only used for the one-day instances of both the attractiveness and the must-go may-go model. Below, Figure 25 and 26 show the results of these instances.

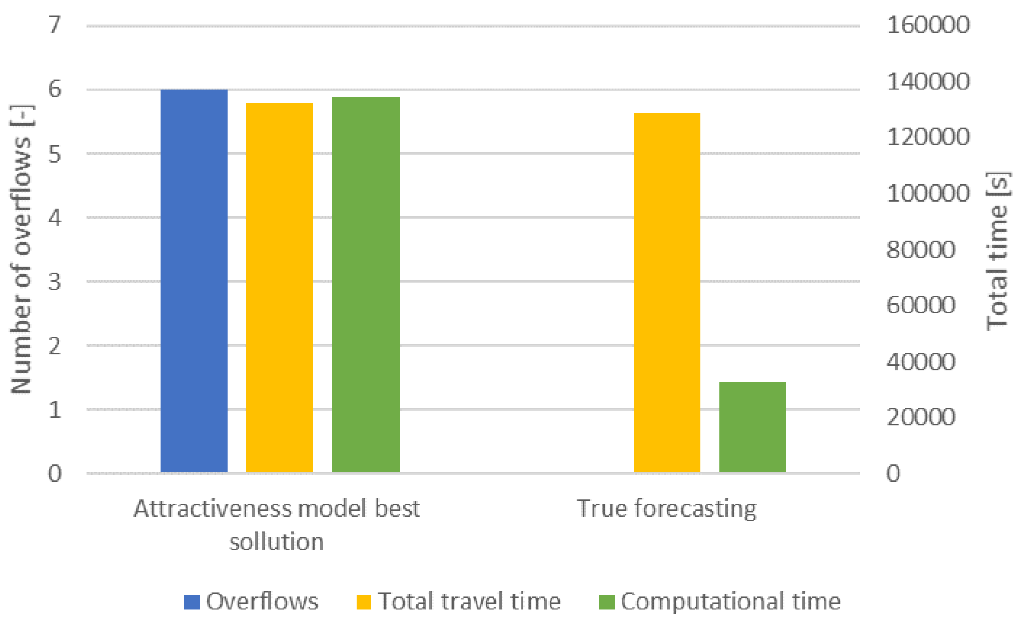

Figure 25: Results model comparison

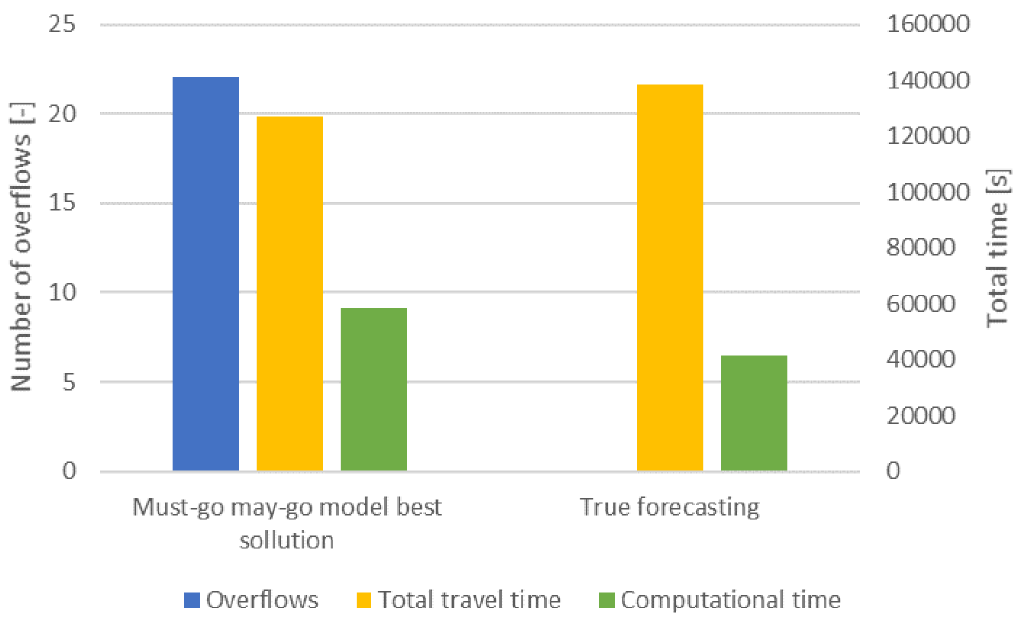

Figure 26: Results model comparison

With this comparison it shows that the best attractiveness instance up to now still can be improved. The comparison with the true forecasting shows that both the total travel time and the computational time can be reduced. Note that this instance of the true forecasting allows no overflows. Since we would allow up to 32 overflows, also the true forecasting model could be optimized to see its full potential. Again, due to large computational times these instances could not be calculated yet. Although the must-go may-go model shows a smaller total travel time, again remember, it is compared to an unoptimized true forecasting model.

5 Conclusion and Recommendations

This section will summarize all the results and discuss the conclusions that are based on them. This will be split up into three parts. First general conclusions, that apply to all models. This will also include the first three sub-questions of this research. Secondly the conclusions for each individual model optimization, which includes sub-question 4. And finally, the conclusions that come from the comparison of the models, which will show the answer to the main research question.

5.1 General conclusions

To start, sub questions 1, 2 and 3 were already discussed and answered in section 2.2 and 3.1, 2.4 and 3.1, and 1.1, 3.2, 3.3 and 3.4 respectively. To summarize, the container selection methods used in literature and practice can be categorized as threshold level based, attractiveness based and must-go may-go based. The KPIs of the waste collection system are total cost, total travelled distance, total travelled time, emitted CO2, number of vehicles used, amount of overflows, vehicle capacity rate or the total amount of collected waste. This research focuses on minimizing the total travel time while keeping the number of overflows in an acceptable range of a baseline of overflows, based on real data. The SWCP can be modelled as a VRP when only threshold levels are taken into account. The moment also a horizon is used and the container selection is integrated into the route optimization, it can be modelled as an IRP. Specific details of the modelled SWCPs are given in section 3.

Besides the sub-questions, other conclusions can be drawn from this research as well.

- There exists a strong negative correlation between the amount of overflows and the total travel time. When the constraints are "loosened" the solution becomes better. As the constraints are loosened, the number of overflows increases.
- The relative increase in overflows is much larger than the relative decrease in total travel time. Depending on the company or regulations, a baseline or target number of overflows can be set in order to minimize the total travel time accordingly.
- The use of more vehicles enlarges the computational time significantly. The fact that the use of only 1 or 2 routes and a network of 151 containers already takes such a long time makes it impossible to do calculations on the whole network (408 containers and possibly 4 routes).
- A start-up period or +/- 3 days was observed for almost any model. Since it is very hard to point down when precisely the steady-state of the smart waste collection system is reached, a visual estimate was used. The collected amount of waste will always oscillate around the deposited amount of waste, which also shows a slight oscillation.

5.2 Conclusions on model optimization

The optimization of each model was done by changing tuning parameters in such a way that the total travel time was minimized and the number of overflows was kept within a reasonable distance from the overflow baseline. Due to the long computational times, only a small set of instances was run. More instances could be calculated in the future to investigate certain details more specifically and possibly even improve on the found solutions.

As for the threshold model, the smallest total travel time of 147097 seconds was found for the instance with a threshold buffer of -1%. This was concluded keeping in mind that with a threshold buffer of -1% the number of overflows was found to be 31. The attractiveness model shows the best results for the instance with a three-day horizon and an upper limit of 100%. Here a total driving time of 132259 seconds was found, with 6 overflows. Finally, the must-go may-go model shows a minimal total travel time of 126828 seconds for the instance with a three-day horizon, a threshold buffer of -5% and an upper limit of 110%, with 22 overflows.

So far, both the attractiveness model and must-go may-go model show the best results for the three-day horizon instances. Strangely, the two-day horizon instances do not show significant improvements when compared to the one-day horizon.

5.3 Conclusions on model comparison

When comparing the models, it is observed that the must-go may-go model obtains the best results when it comes to total travel time. However, both the attractiveness model and the must-go may-go model have room for improvement since the number of overflows is allowed to increase.

5.4 Recommendations

After the literature review, the first recommendation has to be about data. If data is generated, it should be done in such a way that it is realistic, or even better, the use of real data. Smart waste collection is already employed in multiple countries and this data should not be too confidential. The main point of improvement for the models discussed in this research is the computational time. The large computational times limited the depth of research. For further research into the SWCP, the use/development of a heuristic is highly recommendable. Reducing computational time will allow for quicker and therefore more detailed research. The first step for the models in this research would be to be compared to a model which is used in real life, for example, the one of AMCS. When it is proven that the performance of the models of this research are acceptable other details can be investigated as well. A small detail which could be added is the division of overflows in two categories, small and large overflows. Since most overflows are only just above 100% and based on valve movement, it is not sure this directly translates to a physical overflow. To follow up on this research, the influence of each tuning parameter could be further investigated, including the exclusion level. After which, if still present the under-performance of the two-day horizon could be investigated, both for the attractiveness model and the must-go may-go model, as well as the true forecasting model. The latter can be used to show the potential of the forecasting also for a horizon of multiple days. Topics that were not touched upon by this research, but still very interesting to investigate are the influence of the type of network, i.e. rural vs urban and small vs big, to see which model performs best under which conditions.

References

Abdallah, M., Adghim, M., Maraqa, M., and Aldahab, E. (2019). Simulation and optimization of dynamic waste collection routes. *Waste Management and Research*, 37(8):793–802.

Akhtar, M., Hannan, M. A., Begum, R. A., Basri, H., and Scavino, E. (2017). Backtracking search algorithm in CVRP models for efficient solid waste collection and route optimization. *Waste Management*, 61:117–128.

Bueno-Delgado, M. V., Romero-Gázquez, J. L., Jiménez, P., and Pavón-Mariño, P. (2019). Optimal path planning for selective waste collection in smart cities. *Sensors (Switzerland)*, 19(9):1–14.

Catania, V. and Ventura, D. (2014). An approch for monitoring and smart planning of urban solid waste management using smart-M3 platform. *Conference of Open Innovation Association, FRUCT*, pages 24–31.

EuopeanCommissionEnviroment (2019). Waste policy review - environment - european commission. https://ec.europa.eu/environment/waste/target_review.htm.

Hannan, M. A., Akhtar, M., Begum, R. A., Basri, H., Hussain, A., and Scavino, E. (2018). Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm. *Waste Management*, 71:31–41.

Johansson, O. M. (2006). The effect of dynamic scheduling and routing in a solid waste management system. *Waste Management*, 26(8):875–885.

Lozano, Á., Caridad, J., De Paz, J. F., González, G. V., and Bajo, J. (2018). Smart waste collection system with low consumption LoRaWAN nodes and route optimization. *Sensors (Switzerland)*, 18(5):1–24.

Mamun, M. A. A., Hannan, M. A., Hussain, A., and Basri, H. (2016). Theoretical model and implementation of a real time intelligent bin status monitoring system using rule based decision algorithms. *Expert Systems with Applications*, 48:76–88.

Markov, I., Maknoon, Y., and Varone, S. (2016). Inventory routing with non-stationary stochastic demands Inventory routing with non-stationary stochastic demands Transport and Mobility Laboratory École Polytechnique Fédérale de Lausanne Transport and Mobility Laboratory , École Polytechnique Fédérale d. (August).

McLeod, F. and Cherrett, T. (2008). Quantifying the transport impacts of domestic waste collection strategies. *Waste Management*, 28(11):2271–2278.

Mes, M., Schutten, M., and Rivera, A. P. (2014). Inventory routing for dynamic waste collection. *Waste Management*, 34(9):1564–1576.

Nations, U. (2014). *World urbanization prospects*. Statistical Papers - United Nations (Ser. A), Population and Vital Statistics Report. UN.

NDT-ResourceCenter (2006). Accuracy, error, precision, and uncertainty. https://www.nde-ed.org/GeneralResources/ErrorAnalysis/UncertaintyTerms.htm.

of-Economic-and Social-Affairs, U.-N.-D. (2017). World-population-prospects-2017. https://www.un.org/development/desa/en/news/population/world-population-prospects-2017.html.

ORCA-Media (2019). How iot solutions can improve waste management processes. https://www.iotforall.com/smart-iot-solution-assist-legacy-process-management/, http://podorca.com/ .

Papalambrou, A., Karadimas, D., Gialelis, J., and Voyiatzis, A. G. (2015). A versatile scalable smart waste-bin system based on resource-limited embedded devices. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2015-October(September).

Ramos, T. R. P., de Morais, C. S., and Barbosa-Póvoa, A. P. (2018). The smart waste collection routing problem: Alternative operational management approaches. *Expert Systems with Applications*, 103:146–158.

Santos, L., Coutinho-Rodrigues, J., and Current, J. R. (2008). Implementing a multi-vehicle multi-route spatial decision support system for efficient trash collection in Portugal. *Transportation Research Part A: Policy and Practice*, 42(6):922–934.

Tavares, G., Zsigraiova, Z., Semiao, V., and Carvalho, M. (2009). Optimisation of MSW collection routes for minimum fuel consumption using 3D GIS modelling. *Waste Management*, 29(3):1176–1185.

Waste-insight (2017). Omgekeerd inzamelen verandert systeem voor routeplanning en afvalinzameling. https://www.waste-insight.nl/omgekeerd-inzamelen-verandert-systeem-voor-routeplanning-en-inzameling/.

Appendix A - Scientific Paper

Optimizing smart waste collection through container selection

Jelmer van Zeijl, Bilge Atasoy and Rudy Negenborn

Department of Maritime and Transport Technology, Delft University of Technology, The Netherlands

Abstract

The topic of smart waste collection (SWC) is the optimization of the waste collection process using sensor containing waste containers, that can measure and communicate the fill levels. These fill levels can be used to optimize a system based on the chosen key performance indicators (KPI's). This paper focuses on minimizing the total travel time whilst keeping the number of overflows below a baseline, which is based on real data. Furthermore, this paper compares three different container selection methods, using real data obtained from AMCS. The first method is threshold level based, the second attractiveness based and the third one is must-go may-go based. The later is a combination of the first two. Optimization shows promising results for both the three-day horizon instances of the attractiveness and the must-go may-go model. Although the attractiveness has a slightly smaller total travel time, the must-go may-go is more suitable for application due to an acceptable computational time and a smaller number of overflows.

1 Introduction

At the moment, more than half of the world's population lives in urban areas. By 2050, the United Nations expects this amount to have increased to two-thirds of the world's population (Nations, 2014; of-Economic-and Social-Affairs, 2017). Add to this the forecast of population growth from 7.7 billion people now to 9.7 billion in 2050 and a major challenge arises for the cities of the future. By increasing awareness that our waste can have environmental and economical benefits that could be taken advantage of, the interest in waste collection has grown as well. Generally, municipalities are responsible for the complete waste management system, which consists of collecting, transporting, processing, recycling, disposing, and monitoring of the waste materials. The costs of collection and transportation alone accounts for about 70% of the waste management costs (Tavares et al., 2009). These high collection and transportation costs combined with the forecast of increased waste production in cities demand a smart solution to deal with this problem.

The current method in which the vast majority of waste is collected is a static or periodic collection that is using fixed routes and is also revered to as *blind collection*. Here each truck follows specific routes on specific days and picks up all containers along its route. This method often already includes some work of optimization, because some containers are emptied more often than others and routes are matched to a truck's capacity, but there is still a lot of room for improvement. Ramos et al. (2018) shows an example in which on average 10% of all bins on a route is empty, even reaching a maximum of 38% of the bins being empty. Furthermore, approximately 66% of the fill-levels was registered below or equal to 50%.

This paper will discuss the topic of smart waste collection and routing problems. The basic idea of smart waste collection starts with the use and implementation of sensors into the waste containers to measure their fill-levels. Because these sensors can be integrated into the Internet of Things, (IoT) the acquired data-points can be sent to servers where they are stored, processed and used for forecasting, supervision, and finally making smart decisions to optimize the collection of waste. Figure 1 shows that when the fill-levels are known, not all containers need to be visited and routes can be optimized. The optimisation is often based on an objective function, which generally minimizes the total cost and could include travelling costs, labour costs, truck-related expenses, penalties and maximizes profits obtained from the collected waste.

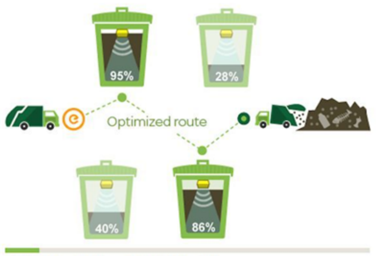

Figure 1: Optimized route Waste-insight (2017)

2 Literature review

2.1 Data

The optimization of routing problems is generally based on the number of trucks available and more importantly on the set of customers, i.e. containers that are selected for the collection. The set of containers ready for collection could in the future be based on the data coming from sensors, that measure and communicate the fill levels of all containers. This data is the foundation on which the route optimization will be built. Therefore, the level of success of route optimization depends on the accuracy of the data. The fill level of waste containers can be obtained in many different ways, i.e. by many different sensors, each sensor measuring other physical properties. Each has different pros and cons. The most discussed type of sensor in literature on the topic of smart waste collection is the volumetric sensor. As the name suggests this sensor measures the volume of waste in the container, also called the fill level. Regardless of the type of sensor that is used, the sensors measure a certain value which will always contain a level of uncertainty, created by different errors. The uncertainty is something which can not be eliminated, i.e. it will always be there. The errors that cause, or define the range of uncertainty can sometimes be minimized but never completely eliminated. In Mamun et al. (2016) an experiment is conducted to find the errors in a set of ultrasonic sensor measurements. During 36 measurements the errors are found to be between -7.8% and +14.4% of the true fill level. Therefore proving that data should be cleaned, and errors should be minimized if possible. Never the less, most articles do not deal with uncertainties, data treatment or mistakes in the measurements, as they should have. Mes et al. (2014) are the only ones that do not directly propose using the measurements obtained by the sensors. They suggest using estimates for the deposit volumes, the waste levels in the containers and the amount of waste overflow. Although the way these estimates are made is not described in their article, the purpose of the estimates is to cope with uncertainties in waste deposits. The lack of detail in this article about the way sensor data is treated and the pure absence of it in the rest of literature articles shows a discrepancy between academic work and reality, taking into account the lack of validation given for the direct use of the data.

Due to the fact that installing hundreds or thousands of sensors is a time consuming and costly process, artificial data is often generated to be able to run test cases in order to prove the success of an optimization method for the SWCP. Using direct data from the sensors is most often not possible as aforementioned, although the data could be based on the actual fill levels of the containers in question. A good example of this is the article of Ramos et al. (2018), they have let the collection team track the fill levels of 3 routes for a time span of 30 days. Due to time windows in between pickups, the fill levels are averaged over this time window to obtain a daily deposit rate. This method captures the general deposit rates in the studied area but is not able to capture and use daily variations in the deposits. Abdallah et al. (2019) take it a step further. A similar approach is used, only now the fill levels are checked daily. The authors even include a field survey in the article. Not all articles need realistic data though, it depends on the goal of the article. When the goal is to test the proposed work in a test case, the data should give a realistic representation of the studied area. A counterexample is shown in Bueno-Delgado et al. (2019), here the fill levels are randomly set to levels between 1 and 100%, and the lack of installed sensors is the only reason given for this assumption. Mes et al. (2014), on the other hand, focuses on parameter tuning and for this reason chooses to make assumptions when generating data, consciously simplifying the tested case.

2.2 Container selection

The general idea of smart waste collection is the use of sensors, that collect and communicate the fill levels of all containers. Based on these fill levels, decisions are made about which containers and via which route they are collected. In the discussed literature, three main methods for container selection are proposed:

1. Based on a pre-determined threshold level of a container
2. Based on the attractiveness of a container
3. Based on both the threshold level and the attractiveness

The use of threshold levels is by far the easiest of all methods and most used in literature. However, a wide variety of it is used in literature. The simplest way is to set the same threshold level, for all containers, equal to a percentage of a bin's capacity and collecting every container whose level is equal or higher than the threshold. This can be improved by giving different groups of containers, different threshold levels. Containers that get large deposits have a lower threshold level and the ones with small deposit rates have high threshold level. Taking it again one step further, Abdallah et al. (2019) and Lozano et al. (2018) include the predicted deposit volume for that day and add it to the fill level already present in the bin and measured by the sensor.

Another method to select containers is based on the attractiveness of a container. This method aims to collect each container on the best day possible.

Ramos et al. (2018) defines a container to be attractive when the fill level is maximized and the transportation costs incurred to collect them are minimized. Both factors are included in the optimization function as well as the penalty for the usage of a certain amount of vehicles. The objective function thus maximized the profit. In other words, it maximizes the amount of collected waste and minimizes the amount of distance travelled and the number of trucks needed for the collection. Although Markov et al. (2016) does not use the word attractiveness, the method used in their article is closest related to the pickup decisions made, based on attractiveness.

The last of the three methods is a combination of the other two. The only article that combines the two methods is Mes et al. (2014). Here containers are labelled as one of the following three categories: 'Must Go', 'May Go' and 'No Go'. The 'Must Go' containers are based on the first method, the one based on threshold levels. Interesting is the fact that Mes et al. (2014) do not set a threshold on the fill level of the container as other articles do (Abdallah et al., 2019; Lozano et al., 2018; Akhtar et al., 2017; Hannan et al., 2018). Instead, Mes et al. (2014) puts a threshold on the number of days in which a container is expected to be full.

2.3 Optimization methods

As routing problems developed from simple travelling salesman problems to the complex vehicle- and inventory routing problem, methods for solving them also developed. Where heuristics provide good to optimal solutions, mathematical programming guarantees to find the optimum solution. However this comes at a price, mathematical programming includes large computational times. This makes them unsuitable for complex problems or problems that need to be solved in a relatively short amount of time, such as the waste collection problem. Heuristics offer quick and good solutions making them better suitable for the waste collection problem. Nowadays, often meta-heuristics are used or even multiple heuristics of meta-heuristics at the same time. In the latter, these heuristics compete for the best solution within a set time limit.

3 Problem description

As aforementioned, the SWCP can be categorized as both a VRP or an IRP. In the case of a VRP, the list of customers (in this case the containers) that will be visited should be known before the route optimization starts. This significantly simplifies the problem and is the case for the threshold discussed in this paper. When the selection of customers is combined with the route optimization we categorize our SWCP's as IRP, only now instead of replenishing the customers, the containers will be emptied. This is the case for the attractiveness model and the must-go may-go model. For all models discussed in this paper, the network consists of 154 nodes, of which 151 are containers, one is the depot and two are used for the terminal. All nodes are located in or around the city of Leeuwarden, the Netherlands. The collection will be carried out by two homogeneous vehicles that both can carry out two routes a day. Each vehicle will start at the terminal, visit a certain set of containers, visit the depot and drive back to the terminal. The chosen KPI's for this research are total travel time and the number of overflows. The total travelling time consists of actual driving times and times spent emptying containers, so-called service time. An overflow occurs when the fill level of a container exceeds the 100%. The objective function of each model formulated to minimize only total travel time. The number of overflows will be recorded for each model and will allow up to a certain baseline. This baseline of overflows is calculated using real container data, obtained from AMCS (a company that develops software to optimize a SWCP, amongst other things) and OMRIN (a waste collection company in charge of the Leeuwarden area). Listed below are the details that are identical to the AMCS/OMRIN case and help to make the models discussed in this paper as realistic as possible:

- Real fill levels for the starting day
- Real daily deposit rates (except for some due to collection in real time)
- Real network of containers, terminal and depot, including locations and volumes
- Realistic travel times calculated using the actual road network
- Same time windows for a subset of containers in the city centre
- Same amount of trucks/routes
- Same volumetric truck capacity
- Same starting times for the trucks
- Same service times for the containers and the depot

Listed below are all assumptions and simplifications that were made in the models and cause deviation from the AMCS/OMRIN case and reality:

- No breaks at the terminal from 12:45 to 13:00
- No custom orders besides the 408 containers
- No visits to Harlingen twice a week, with 1 truck
- No use of an additional depot in the city of Franeker
- Collection during every day of the week (Saterday and Sunday as well)
- Smaller network, 151 in stead of 408
- No deposits between measurement and collection

4 Mathematical models

4.1 Threshold model

The detailed mathematical model showed in this paper represents the must-go may-go model. This differs from the threshold model in multiple ways. The main difference between the threshold model and the must-go may-go model is that the threshold model selects containers separately from the route optimization. The fill levels of the containers are compared with specific thresholds levels, if a fill level is above its threshold level it will be added to the list called *visit*. The constraints (1) to (7) from the must-go may-go model can therefore be replaced by the following:

$$\sum_{j\in nid}\sum_{k\in K} x_{i,j,k}=1 \qquad \forall i \in visit \qquad (1)$$

Constraint (13) of the must-go may-go model is replaced by:

$$x_{408,409,k}=1 \qquad \forall k \in K \qquad (6)$$

Constraint (21) and (22) of the must-go may-go model are replaced by:

$$w_{j,k}\geq w_{i,k}+cc_j-M*(1-x_{i,j,k})$$
$$\forall i \in nid, \forall j \in nid, \forall k \in range(K) \qquad (16)$$

$$w_{j,k}\leq w_{i,k}+cc_j+M*(1-x_{i,j,k})$$
$$\forall i \in nid, \forall j \in nid, \forall k \in range(K) \qquad (17)$$

4.2 Attractiveness model

The detailed mathematical model showed in this paper represents the must-go may-go model. This differs only slightly from the attractiveness model. Only constraint (7) needs to be removed from the must-go may-go model to obtain the attractiveness model. This removes the specific threshold levels, only the overall upper limit, constraint (5) of the must-go may-go model is used in the attractiveness model to force the collection of containers.

4.3 Must-go may-go model

The must-go may-go model is the largest model of all the models discussed in this paper and therefore it is displayed in more detail. Note that the objective function and all constraint who were not mentioned in the other models, apply to all three the model.

Objective function:

$$\text{MIN} \sum_{i\in nid}\sum_{j\in nid}\sum_{k\in K}\sum_{day\in days} \text{x}_{i,j,k,day} * \text{time\_matrix}_{i,j} + \text{st}_i * \text{x}_{i,j,k,day}$$

Constraints:
In ascending order, constraint (1) copies the fill levels of the total fill level list to the first day of the horizon. Constraints (2-4) fill the rest of the fill levels within the horizon with the influence of the collection of a container. Constraint (5) prevents overflows, constraint (6) reduces computational time by excluding containers with low fill levels. Constraint (7) forces containers to be collected if their fill level exceeds their threshold level. Constraint (8) will prevent arcs from and to the same node. Constraints (9-13) forces each route to start at node 407, visits node 408 if waste was collected and end at node 409. Note again that node 407 and 409 are two nodes representing the same terminal and node 408 is the deposit site. Constraint (14) ensures flow conservation at each node. Constraints (15-19) prevent sub tours by the numbering of nodes and only allowing the visit of nodes with higher numbers. Constraint (20) sets the load at start point equal to 0. Constraints (21 and 22) ensure load continuity and constraint (23) limits the load of each vehicle. Constrains (24 and 25) ensure visit time continuity and finally, constraint (26) makes sure all nodes in early are collected before 9.00 AM. Note that in constraints (21), (22), (24) and (25) the big M method is used. The displayed in the constraint represents the number 100000.

$$hfl_{i,0} = total\_fill\_level\_list_{i,0} \qquad \forall i \in nid \tag{1}$$

$$hfl_{i,1} = hfl_{i,0} * (1 - \sum_{i \in nid} \sum_{k \in K} x_{i,j,k,0}) + add_i \qquad \forall i \in nid, \tag{2}$$

$$hfl_{i,2} = hfl_{i,1} * (1 - \sum_{i \in nid} \sum_{k \in K} x_{i,j,k,1}) + add_i \qquad \forall i \in nid, \tag{3}$$

$$hfl_{i,2} = hfl_{i,1} * (1 - \sum_{i \in nid} \sum_{k \in K} x_{i,j,k,1}) + add_i \qquad \forall i \in nid, \tag{4}$$

$$hfl_{i,1}\ leq100 \qquad \forall i \in nid, \forall day \in days \backslash 0 \tag{5}$$

$$hfl_{i,day} \geq 40 * \sum_{j \in nid} \sum_{k \in K} x_{i,j,k,day} \qquad \forall i \in nid, \forall day \in days \tag{6}$$

$$hfl_{i,0} \leq threshold\_levels_i + 100 * \sum_{j \in nid} \sum_{k \in K} x_{i,j,k,0} \qquad \forall i \in ned \tag{7}$$

$$x_{i,i,k,day} = 0 \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{8}$$

$$\sum_{j \in nid} x_{407,j,k,day} = 1 \qquad \forall k \in K, \forall day \in days \tag{9}$$

$$\sum_{i \in nid} x_{i,407,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{10}$$

$$\sum_{i \in nid} x_{i,409,k,day} = 1 \qquad \forall k \in K, \forall day \in days \tag{11}$$

$$\sum_{j \in nid} x_{409,j,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{12}$$

$$\sum_{i \in nid} w_{i,k} * (1 - x_{408,409,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{13}$$

$$\sum_{j \in nid} x_{i,j,k,day} - \sum_{j \in nid} x_{j,i,k,day} = 0 \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{14}$$

$$se_{407,k,day} = 1 \qquad \forall k \in K, \forall day \in days \tag{15}$$

$$se_{i,k,day} \geq 2 \qquad \forall i \in nid \backslash 407, \forall k \in K, \forall day \in days \tag{16}$$

$$se_{i,k,day} \leq |nid| \qquad \forall i \in nid \backslash 407, \forall k \in K, \forall day \in day \tag{17}$$

$$se_{i,k,day} - se_{j,k,day} + |nid| * x_{i,j,k,day} \leq (|nid| - 1) \qquad \forall i \in nid \backslash \text{|nid|-1}, \forall j \in nid \backslash 407 \quad \forall k \in K, \forall day \in days \tag{18}$$

$$se_{i,k,day} - se_{j,k,day} + |nid| * (1 - x_{i,j,k,day}) \geq -1 \qquad \forall i \in nid \backslash \text{|nid|-1}, \forall j \in nid \backslash 407 \quad \forall k \in K, \forall day \in days \tag{19}$$

$$w_{407,k,day} = 0 \qquad \forall k \in K, \forall day \in days \tag{20}$$

$$w_{j,k,day} \geq w_{i,k,day} + (hfl_j/100) * cv_j - M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in range(K) \quad \forall day \in days \tag{21}$$

$$w_{j,k,day} \leq w_{i,k,day} + (hfl_j/100) * cv_j + M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in range(K) \; \forall day \in days \tag{22}$$

$$w_{i,k,day} <= Q \qquad \forall i \in nid, \forall k \in K, \forall day \in days \tag{23}$$

$$u_{j,k,day} \geq u_{i,k,day} + st_i + time\_matrix_{i,j} - M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{24}$$

$$u_{j,k,day} \leq u_{i,k,day} + st_i + time\_matrix_{i,j} + M * (1 - x_{i,j,k,day}) \qquad \forall i \in nid, \forall j \in nid, \forall k \in K \tag{25}$$

$$u_{i,k,day} = 32400 \qquad \forall i \in early, \forall k \in K \tag{26}$$

5 Results

This chapter will show results for each of the developed models, as well as the comparison between the models. Note again that the main KPIs are the total travel time and the number of overflows. The later needs to be below the set baseline.

5.1 Threshold model

Since for the threshold model, the fill levels can be calculated without finishing the entire route optimization the results for many different buffer values could be calculated. To get a reference for all other parameters a first instance was calculated for a threshold buffer of 5%, this is shown as 'Threshold 1' in Table 1. A threshold buffer of -1% was found to fit the baseline of 32 overflows best and also shows a lower total travel time. This instance is shown in Table 1 as 'Threshold 2'. All threshold results show a strong and clear negative correlation between the total travel time and the number of overflows.

5.2 Attractiveness model

To optimize the attractiveness model, two parameters will be changed that will influence the selection of containers, namely, the length of the horizon and the upper limit. The first three calculated instances all have an upper limit of 100% and one-, two- and three-day horizons. These are shown as 'Attractiveness 1', 'Attractiveness 2' and 'Attractiveness 4'. Note that the one- and three-day horizon are competitive, whilst the two-day horizon underperforms. However, the one-day horizon already exceeds the overflow baseline and will not be further minimized. The three-day horizon is further optimized with an upper limit of 110% and shown as 'Attractiveness 5'. The two-day horizon is also optimized using a 110% upper limit and shown as 'Attractiveness 3', but it has a significantly higher total travel time than the three-day horizon instance, which shows the lowest total travel time. Note however that it does exceed the overflow baseline and has a computational time which is larger than the actual time needed to collect the containers in real life.

5.3 Must-go may-go model

'Must-go may-go 1' and 'Must-go may-go 2' show the initial and optimized instances for a one-day horizon. More interesting to see is that again, the two-day horizon is performing worse than the one- and three-day horizon. Although the must-go may-go model never shows a total travel time lower than 'Attractiveness 5', it is however very competitive as 'Must-go may-go 6' shows. This instance has a three-day horizon, a -7% threshold buffer and an upper limit of 110%. The true benefit of this latest instance is the smaller computational time, as well as a drop in overflows of 33%.

5.4 Model comparison

As already mentioned. Although the optimized attractiveness model with a three-day horizon shows the lowest total travel time, it is not applicable to reality due to the mathematical way in which it is solved. If a mathematical solver is used, the optimized three-day instance of the must-go may-go model shows an almost equal total travel time with a significant reduction in both computational time and overflows.

Table 1: Results all models

	Total time [s]	Total volume [L]	Overflows [-]	Run time [s]	Nr. nodes [-]
Threshold 1.1	158717	1142441	5	39	283
Threshold 1.2	156354	1148279	11	126	272
Threshold 1.3	147097	1139260	31	176	253
Threshold 1.4	145436	1125687	69	65	238
Attractiveness 1.1	133129	1116266	35	62514	243
Attractiveness 1.2	136419	1107914	28	20897	244
Attractiveness 1.3	134551	1095706	13	47436	459
Attractiveness 2.1	141952	1138836	9	44456	282
Attractiveness 2.2	133083	1127281	27	70323	260
Attractiveness 2.3	128585	1075896	39	35591	248
Attractiveness 3.1	132259	1106995	6	134255	281
Attractiveness 3.2	123868	1052929	39	165932	255
Must-go may-go 1.1	135115	1102161	7	2347	258
Must-go may-go 1.2	134498	1084729	16	29265	252
Must-go may-go 1.3	131647	1107676	33	6676	242
Must-go may-go 2.1	147589	1137935	2	84235	287
Must-go may-go 2.2	131162	1191881	24	11200	231
Must-go may-go 2.3	139786	1134688	28	48545	262
Must-go may-go 3.1	137379	1100168	10	76732	283
Must-go may-go 3.2	126828	1069053	22	58444	236
Must-go may-go 3.3	129456	1130697	31	66777	271

6 Conclusions and Future research directions

The main conclusion that can be drawn from this research is that the 3 day horizon instance of the must-go may-go model shows the best result while staying below the overflow baseline. Note that the attractiveness model shows potential to outperform the must-go may-go model, however these instances still need to be calculated. Furthermore, the results can help decision makers choose the right method for their SWCP. For example, the computational times of the attractiveness model are significantly larger than those of the must-go may-go model which could be an important factor if time for calculations is limited. More general conclusions are listed below:

- There exists a strong negative correlation between the amount of overflows and the total travel time. When the constraints are "loosened" the solution becomes better. Note that this is most often the case, but there are exceptions where the number of overflows increases as well as the total travel time.
- The relative increase in overflows is much larger than the relative decrease in total travel time. Depending on the company or regulations, a baseline or target number of overflows can be set in order to minimize the total travel time accordingly. But preventing overflows comes at a relatively small cost.
- The use of more vehicles enlarges the computational time significantly. The fact that the use of only 1 or 2 routes and a network of 151 containers already takes such a long time makes it impossible to do calculations on the whole network (408 containers and possibly 4 routes).
- A start-up period or +/- 3 days was observed for all the models. Since it is very hard to point down when precisely the steady-state of the smart waste collection system is reached, a visual estimate was used. The collected amount of waste will always oscillate around the deposited amount of waste, which also shows a slight oscillation.

The first recommendation, for further research into this topic, has to be about data. If data is generated,it should be done in such a way that it is realistic, or even better, the of real data. SWC is already employed in multiple countries and this data should not be too confidential. Secondly, the computational times turned out to be larger than expected. These large computational times limited the depth of this research. For further research into the SWCP, the use/development of a heuristic is highly recommendable. Reducing computational time will allow for quicker and therefore more detailed research.
Finally, when computational times are reduced and larger networks can be investigated, a comparison to a real life SWCP could be made. When it is proven that the performance of the models of this research are acceptable other details can be investigated as well. To follow up on this research, the influence of each tuning parameter could be further investigated, including the exclusion level.

References

Abdallah, M., Adghim, M., Maraqa, M., and Aldahab, E. (2019). Simulation and optimization of dynamic waste collection routes. *Waste Management and Research*, 37(8):793–802.

Akhtar, M., Hannan, M. A., Begum, R. A., Basri, H., and Scavino, E. (2017). Backtracking search algorithm in CVRP models for efficient solid waste collection and route optimization. *Waste Management*, 61:117–128.

Bueno-Delgado, M. V., Romero-Gázquez, J. L., Jiménez, P., and Pavón-Mariño, P. (2019). Optimal path planning for selective waste collection in smart cities. *Sensors (Switzerland)*, 19(9):1–14.

Hannan, M. A., Akhtar, M., Begum, R. A., Basri, H., Hussain, A., and Scavino, E. (2018). Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm. *Waste Management*, 71:31–41.

Lozano, Á., Caridad, J., De Paz, J. F., González, G. V., and Bajo, J. (2018). Smart waste collection system with low consumption LoRaWAN nodes and route optimization. *Sensors (Switzerland)*, 18(5):1–24.

Mamun, M. A. A., Hannan, M. A., Hussain, A., and Basri, H. (2016). Theoretical model and implementation of a real time intelligent bin status monitoring system using rule based decision algorithms. *Expert Systems with Applications*, 48:76–88.

Markov, I., Maknoon, Y., and Varone, S. (2016). Inventory routing with non-stationary stochastic demands Inventory routing with non-stationary stochastic demands Transport and Mobility Laboratory École Polytechnique Fédérale de Lausanne Transport and Mobility Laboratory , École Polytechnique Fédérale d. (August).

Mes, M., Schutten, M., and Rivera, A. P. (2014). Inventory routing for dynamic waste collection. *Waste Management*, 34(9):1564–1576.

Nations, U. (2014). *World urbanization prospects*. Statistical Papers - United Nations (Ser. A), Population and Vital Statistics Report. UN.

of-Economic-and Social-Affairs, U.-N.-D. (2017). World-population-prospects-2017. https://www.un.org/development/desa/en/news/population/world-population-prospects-2017.html.

Ramos, T. R. P., de Morais, C. S., and Barbosa-Póvoa, A. P. (2018). The smart waste collection routing problem: Alternative operational management approaches. *Expert Systems with Applications*, 103:146–158.

Tavares, G., Zsigraiova, Z., Semiao, V., and Carvalho, M. (2009). Optimisation of MSW collection routes for minimum fuel consumption using 3D GIS modelling. *Waste Management*, 29(3):1176–1185.

Waste-insight (2017). Omgekeerd inzamelen verandert systeem voor routeplanning en afvalinzameling. https://www.waste-insight.nl/omgekeerd-inzamelen-verandert-systeem-voor/routeplanning-en-inzameling/.

Appendix B - Experimental results

Threshold model

Table 19: Results threshold model
Threshold gab = 10%

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	10633	97351	0	2	18
Day 5	18418	123214	0	6	29
Day 6	12768	86740	0	2	25
Day 7	14743	121976	1	5	25
Day 8	12749	75737	0	2	21
Day 9	10226	62642	0	2	19
Day 10	11207	91057	1	2	22
Day 11	12715	100509	2	3	28
Day 12	19080	141872	1	9	31
Day 13	11365	73940	0	2	21
Day 14	13820	76135	0	2	22
Day 15	10993	91268	0	2	22
Sum	158717	1142441	5	39	283

Table 20: Results threshold model
Threshold gab = 5%

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	11932	104031	2	8	23
Day 5	17718	114626	0	23	26
Day 6	11842	97176	0	11	27
Day 7	15084	119680	1	10	23
Day 8	13318	97088	2	7	25
Day 9	12209	59543	1	2	17
Day 10	10816	92238	2	6	20
Day 11	11460	78591	1	10	21
Day 12	12586	96501	1	7	26
Day 13	14468	117643	1	24	22
Day 14	12110	82292	0	8	23
Day 15	12811	88870	0	10	19
Sum	156354	1148279	11	126	272

Table 21: Results threshold model
Threshold gab = -1%

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	11466	93458	3	15	23
Day 5	11347	95732	2	7	19
Day 6	15580	106279	2	26	22
Day 7	10908	89863	0	13	22
Day 8	13496	112941	4	26	23
Day 9	12094	94010	3	11	20
Day 10	12443	82681	5	10	21
Day 11	9929	83395	4	13	16
Day 12	11942	87701	0	14	22
Day 13	9226	67447	2	6	17
Day 14	15272	131235	2	27	24
Day 15	13394	94518	4	8	24
Sum	147097	1139260	31	176	253

Table 22: Results threshold model
Threshold gab = -5%

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	16487	121327	8	15	26
Day 5	10019	85454	4	3	16
Day 6	12205	98940	7	3	19
Day 7	12286	91297	5	5	22
Day 8	13229	112708	6	7	22
Day 9	11859	86024	3	4	21
Day 10	12090	100904	7	3	19
Day 11	9779	73853	9	3	18
Day 12	11452	95606	4	4	18
Day 13	9342	60481	4	5	14
Day 14	13052	87684	4	5	21
Day 15	13636	111409	8	8	22
Sum	145436	1125687	69	65	238

Attractiveness model

Note that the days displayed in yellow are days with an exceptionally high computational run time. This is due to the use of two vehicles on these days, increasing the complexity of the model for that day significantly. The large computational times are mostly due to the fact that the total volume is larger than the vehicle capacity and two vehicles are necessary, but not always.

Table 23: Results attractiveness
Horizon length = 1 day, exclusion level = 60%, upper limit = 100%.

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	12974	102894	2	220	25
Day 5	11064	102503	1	347	20
Day 6	10257	89226	2	244	17
Day 7	9475	78039	4	162	19
Day 8	13917	130897	2	44410	27
Day 9	9640	69463	6	84	17
Day 10	14511	115196	5	3677	23
Day 11	9805	88105	1	104	17
Day 12	8719	72672	3	189	18
Day 13	7245	54700	2	133	13
Day 14	11989	101977	4	1518	25
Day 15	13533	110594	3	11426	22
Sum	133129	1116266	35	62514	243

Table 24: Results attractiveness
Horizon length = 1 day, exclusion level = 60%, upper limit = 99%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	10534	94910	2	256	23
Day 5	10523	95208	2	294	19
Day 6	11681	102692	1	364	21
Day 7	9905	82776	3	228	20
Day 8	13595	116851	3	794	24
Day 9	10169	68087	5	277	17
Day 10	14392	114407	3	597	23
Day 11	9972	88504	1	256	17
Day 12	9698	67821	1	348	17
Day 13	6576	49111	3	282	12
Day 14	15401	119731	2	16559	29
Day 15	13973	107816	2	642	22
Sum	136419	1107914	28	20897	244

Table 25: Results attractiveness
Horizon length = 1 day, exclusion level = 60%, upper limit = 95%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	10106	94239	0	20	228
Day 5	10604	71892	0	306	19
Day 6	14069	114161	0	6386	25
Day 7	10013	83356	2	557	21
Day 8	14489	122793	2	5589	26
Day 9	10493	78599	3	404	17
Day 10	11312	81870	1	1004	21
Day 11	9106	86193	0	293	17
Day 12	11354	87778	2	339	23
Day 13	8646	69606	1	472	18
Day 14	14068	126808	1	31835	24
Day 15	10291	78411	1	231	20
Sum	134551	1095706	13	47436	459

Table 26: Results attractiveness
Horizon length = 2 days , exclusion level = 60%, upper limit = 100%.

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	10354	86882	0	377	19
Day 5	14952	109818	0	3513	25
Day 6	11960	94436	0	495	28
Day 7	7419	83414	3	318	15
Day 8	10872	94578	0	1693	25
Day 9	13554	97105	0	437	27
Day 10	12171	98449	0	382	24
Day 11	7481	42225	2	456	12
Day 12	16218	139448	1	9122	35
Day 13	8403	60996	1	208	12
Day 14	11443	90755	2	2134	25
Day 15	17125	140730	0	25321	35
Sum	141952	1138836	9	44456	282

Table 27: Results attractiveness (ASUS PC)
Horizon length = 2 days, exclusion level = 60%, upper limit = 110%.

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	13323	120036	2	65415	26
Day 5	14985	99311	1	462	27
Day 6	8884	86919	1	342	18
Day 7	9447	72326	6	245	18
Day 8	15629	145571	2	1721	32
Day 9	10472	85974	4	308	19
Day 10	9275	71091	3	331	18
Day 11	9366	89910	1	274	18
Day 12	11344	93523	3	275	25
Day 13	9852	68737	2	160	17
Day 14	9008	88154	1	321	19
Day 15	11498	105729	1	469	23
Sum	133083	1127281	27	70323	260

Table 28: Results attractiveness (ASUS PC)
Horizon length = 2 days, exclusion level = 60%, upper limit = 111%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9938	87278	2	1780	22
Day 5	11032	102479	5	2639	22
Day 6	10736	102222	1	1260	21
Day 7	9898	84998	7	997	21
Day 8	15103	131243	2	19784	28
Day 9	10328	87740	4	1611	19
Day 10	10233	74733	5	1074	19
Day 11	10612	96488	1	897	20
Day 12	12561	97997	2	706	26
Day 13	7450	51202	2	1676	13
Day 14	10400	72900	2	1176	19
Day 15	10294	86616	6	1991	18
Sum	128585	1075896	39	35591	248

Table 29: Results attractiveness
Horizon length = 3 days, exclusion level = 60%, upper limit = 100%.

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	10681	96949	1	2803	26
Day 5	12502	83177	0	2709	24
Day 6	7627	63616	0	12834	15
Day 7	11938	106033	1	28352	25
Day 8	11550	105005	1	41349	29
Day 9	12083	108327	1	28352	25
Day 10	7467	51708	0	1876	14
Day 11	11445	89161	1	3538	22
Day 12	13475	130690	1	18039	31
Day 13	8679	65240	0	3006	18
Day 14	12434	110761	0	3618	27
Day 15	12378	96328	0	5508	25
Sum	132259	1106995	6	134255	281 height

The Figure 30 shows a combination of two colors for the 13th day. The yellow color is still used to mark the day for using multiple vehicles, explaining the very high computational time. The orange is used to indicate that this day was run for a two day horizon, instead of a three day horizon. This was decided after the run with a three day horizon did not find a solution after more than 100.000 seconds. Shortening the rolling horizon showed to decrease the computational time significantly and a solution was found. However, the computational time is still very high with respect to the other days. Note that this was also used in Table 37.

Table 30: Results attractiveness
Horizon length = 3 days, exclusion level = 60%, upper limit = 110%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9856	96451	1	1604	25
Day 5	8726	78544	6	8552	17
Day 6	13853	105798	1	22237	28
Day 7	11139	105006	2	6672	24
Day 8	8477	86621	3	13578	18
Day 9	11806	73313	0	5918	22
Day 10	12226	99579	3	2736	26
Day 11	5484	43508	3	5094	11
Day 12	7898	70231	9	9173	15
Day 13	13865	111665	2	41727	29
Day 14	7566	79496	6	41477	15
Day 15	12972	102717	3	7164	25
Sum	123868	1052929	39	165932	255

Must-go may-go model

Table 31: Results must-go may-go
Horizon length = 1 days, threshold gab = 5% , exclusion level = 60%, upper limit = 100%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	10215	93303	0	156	20
Day 5	13095	103791	0	155	23
Day 6	11593	86529	0	153	24
Day 7	8995	93392	2	209	20
Day 8	13309	105101	1	159	27
Day 9	10473	76584	1	159	17
Day 10	10443	93425	1	223	20
Day 11	10100	74897	1	198	20
Day 12	12840	100053	1	267	27
Day 13	10054	86037	0	169	18
Day 14	12396	99515	0	300	23
Day 15	11602	89534	0	199	19
Sum	135115	1102161	7	2347	258

Table 32: Results must-go may-go
Horizon length = 1 days, threshold gab = 2% , exclusion level = 60%, upper limit = 100%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9322	95199	1	309	20
Day 5	12723	105623	0	1487	23
Day 6	11123	86804	0	451	23
Day 7	9869	74955	3	301	19
Day 8	15275	123433	3	22668	26
Day 9	10724	69638	3	385	19
Day 10	8735	82415	2	506	16
Day 11	10606	95208	0	766	19
Day 12	12717	99244	1	600	26
Day 13	10334	96086	0	416	20
Day 14	11670	89427	1	672	23
Day 15	11400	66697	2	704	18
Sum	134498	1084729	16	29265	252

Table 33: Results must-go may-go
Horizon length = 1 day, threshold buffer = -1%, exclusion level = 60%, upper limit = 100%.

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9915	89702	2	156	22
Day 5	10873	99555	1	160	20
Day 6	12510	106012	1	273	22
Day 7	10304	86826	5	415	20
Day 8	12538	118654	3	617	21
Day 9	9321	63550	7	468	16
Day 10	12632	105644	6	224	21
Day 11	8686	88729	1	179	17
Day 12	11275	83226	2	514	21
Day 13	8452	64152	2	196	16
Day 14	13150	110398	2	2919	23
Day 15	11991	91228	1	555	23
Sum	131647	1107676	33	6676	242

Table 34: Results must-go may-go (ASUS PC)
Horizon length = 2 days, threshold gab = 5%, exclusion level = 60%, upper limit = 100%.

	Total time [s]	**Travel volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	8923	72850	0	492	16
Day 5	16231	119589	0	59023	28
Day 6	12570	100641	0	102	30
Day 7	9631	104280	0	238	21
Day 8	10677	71822	0	390	20
Day 9	11807	77942	1	531	23
Day 10	14012	107296	0	9416	26
Day 11	9427	56397	1	131	17
Day 12	14620	121451	0	7561	30
Day 13	9979	83513	0	195	18
Day 14	12988	99309	0	891	29
Day 15	16724	122845	0	5265	29
Sum	147589	1137935	2	84235	287

Table 35: Results must-go may-go (ASUS PC)
Horizon length = 2 days, threshold gab = -5%, exclusion level = 60%, upper limit = 110%

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	11194	193952	1	1465	22
Day 5	11541	84316	1	131	23
Day 6	11440	105374	1	554	23
Day 7	9870	76358	6	185	19
Day 8	15631	141959	3	3835	31
Day 9	10090	85302	3	166	19
Day 10	9475	69632	4	284	18
Day 11	9005	90177	1	349	18
Day 12	11107	90037	2	309	24
Day 13	8071	59452	1	153	15
Day 14	9973	87889	1	245	19
Day 15	13765	107433	0	3524	23
Sum	131162	1191881	24	11200	231

Table 36: Results must-go may-go (ASUS PC)
Horizon length = 2 days, threshold gab = -7%, exclusion level = 60%, upper limit = 110%

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9160	80323	4	4291	20
Day 5	16833	131387	1	18184	30
Day 6	10485	91669	1	1109	19
Day 7	9097	71686	5	778	18
Day 8	15824	140830	3	13855	31
Day 9	10137	86064	4	738	19
Day 10	11453	74061	5	811	19
Day 11	9599	84325	1	1124	17
Day 12	12183	101713	2	1222	27
Day 13	10562	70918	1	961	18
Day 14	10968	94279	1	2207	21
Day 15	13485	107433	0	3265	23
Sum	139786	1134688	28	48545	262
hline					

Table 37: Results must-go may-go
Horizon length = 3 days, threshold gab = 5%, exclusion level = 60%, upper limit = 100%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9430	60417	0	3411	17
Day 5	13925	95099	0	9429	27
Day 6	10289	94078	0	2791	23
Day 7	11149	104344	0	5431	26
Day 8	13531	100455	0	6670	27
Day 9	12060	89216	1	3201	22
Day 10	9615	73405	1	1986	23
Day 11	10443	88039	1	1808	22
Day 12	10549	104592	1	27608	24
Day 13	10600	95122	6	7848	21
Day 14	12551	92343	0	2928	27
Day 15	13237	103058	0	3621	24
Sum	137379	1100168	10	76732	283

Table 38: Results must-go may-go (ASUS PC)
Horizon length = 3 days, threshold gab = -5%, exclusion level = 60%, upper limit = 110%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9724	78202	0	3672	17
Day 5	6964	36215	3	2023	10
Day 6	17047	148096	0	3490	42
Day 7	5295	47597	4	4008	9
Day 8	13627	142036	1	1514	33
Day 9	9436	74048	3	8503	17
Day 10	8299	58458	5	8925	15
Day 11	14484	118154	1	8757	31
Day 12	10479	95635	1	3781	23
Day 13	4629	51405	1	5522	9
Day 14	14039	112953	3	2985	30
Day 15	12805	106254	0	5264	31
Sum	126828	1069053	22	58444	236

Table 39: Results must-go may-go (ASUS PC)
Horizon length = 3 days, threshold gab = -7%, exclusion level = 60%, upper limit = 110%.

	Total time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	11341	101752	5	1851	27
Day 5	16728	131996	1	6929	30
Day 6	11599	83757	1	1869	23
Day 7	8270	85419	3	3254	19
Day 8	10407	103258	3	4940	23
Day 9	7644	80074	7	4046	18
Day 10	12793	105831	3	5264	27
Day 11	11639	92673	1	3387	26
Day 12	6372	43288	2	2487	12
Day 13	14172	101858	0	22981	28
Day 14	8174	99655	2	6739	16
Day 15	10317	101136	3	3030	22
Sum	129456	1130697	31	66777	271
hline					

True forecasting model

Table 40: Results attractiveness true forecasting
Horizon length = 1 day, exclusion level = 60%, exclusion level = 100%.

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	11959	103866	0	164	26
Day 5	10311	90855	0	70	18
Day 6	10958	98716	0	148	21
Day 7	10614	88594	0	64	22
Day 8	13756	122511	0	29928	26
Day 9	10561	90391	0	326	19
Day 10	11957	86541	0	288	22
Day 11	7467	67858	0	397	13
Day 12	11419	92816	0	412	24
Day 13	10902	84248	0	231	22
Day 14	10140	68874	0	380	17
Day 15	8805	101419	0	334	15
Sum	128849	1096689	0	32742	245

Table 41: Results must-go may-go true forecasting
Horizon length = 1 day, threshold gab = 5%, exclusion level = 60%, upper limit = 100%.

	Travel time [s]	**Total volume [L]**	**Overflows [-]**	**Run time [s]**	**Nr. nodes [-]**
Day 4	9347	85644	0	310	18
Day 5	15670	113717	0	5136	26
Day 6	10250	79692	0	225	22
Day 7	9744	103224	0	156	23
Day 8	15085	124723	0	33597	28
Day 9	9778	55330	0	156	16
Day 10	11812	94718	0	256	21
Day 11	10399	71712	0	163	21
Day 12	14717	107626	0	979	29
Day 13	9611	80025	0	149	17
Day 14	11023	91043	0	110	20
Day 15	11067	89162	0	208	19
Sum	138503	1096616	0	41445	260