Bachelor Thesis Industrial Engineering and Management

Optimizing Route Planning for Connected Green's Customers

Faculty: Behavioural, Management and Social Sciences (BMS)

Author:

J. Kortekaas (Jordi)

S2376997

20-09-2023

Supervisors University of Twente:

1. Dr.ir. E.D. Lalla (Eduardo)
2. Dr.ir. L.L.M van der Wegen (Leo)

Supervisor Connected Green:

H. Schaap (Hans)

UNIVERSITY OF TWENTE.

CONNECTED GREEN

CONTENTS

ACKNOWLEDGEMENT

Dear reader,

You are about the read my bachelor thesis that will complete my Bachelor Industrial Engineering and Management (IEM) at the University of Twente. The research for this thesis has been conducted at Connected Green. However, before you proceed with reading the thesis I would like the people who have supported and guided me through the process of this research.

First, I would like to thank Hans Schaap, my company supervisor, for the opportunity to conduct the research for my bachelor thesis at Connected Green and for the guidance he has given me throughout the process. Furthermore, I would like to thank the employees of Curious Inc. (the parent company of Connected Green) for the warm welcome they gave me when I first started my research.

Secondly, I would like to thank dr. Lalla, who was my supervisor of the University of Twente. From the start of creating the project plan until the end of the thesis, he has been supporting and guiding me. The feedback dr. Lalla provided me throughout the process, has given me a different perspective which has helped in writing my thesis. Furthermore, I want to thank dr. van der Wegen for being my second supervisor. Finally, I would like to thank my friends and family for their support during the research.

Kind regards,

Jordi Kortekaas

Enschede, August 2023

MANAGEMENT SUMMARY

This research has been performed at Connected Green, a Curious Inc. company located in Hengelo. Connected Green focuses on giving data insight into green projects, such as planting new plants and trees. It does this by selling soil moisture sensors which are planted near the roots of these plants or trees. These sensors send the measured moisture levels to a dashboard. This dashboard indicates which plants or trees currently need the most attention in terms of watering. The customers of Connected Green use these insights to monitor their green projects and plan irrigation routes when the sensors indicate the plants or trees are too dry. These irrigation routes are driven by cars or tractors with large water tanks. As these routes are often quite costly, it is valuable to have a planning strategy which produces irrigation routes that are close to optimal. However, the current planning strategies employed by the customers of Connected Green are far from optimal. To solve this problem, this research focuses on creating a solution model that will automate the irrigation route planning for the customers of Connected Green. Therefore, the main research question of this thesis is formulated as follows:

What is the most effective approach to develop an optimal route planning model, and how can it be leveraged by Connected Green to minimize the costs incurred by their customers?

To develop a solution model for the customers, it is first necessary to understand the situation at both the customers and Connected Green. To do this, a contextual analysis has been performed. In discussions with Connected Green's customers and employees and the information from the dashboard, a better insight into the current route planning strategies, constraints, customer requirements, and Key Performance Indicators is achieved. The current route planning strategies were identified to be far from optimal, ranging from strategies which visit every sensor regardless of needing irrigation to strategies which start at the most northern sensor and by driving south visit each sensor closest to the current sensor until all sensors are visited. The constraining factors for the model were identified to be road accessibility and water tank capacity. Where road accessibility means that some vehicles are not allowed or are unable to drive on certain roads, for example, a tractor which is too wide to drive on a narrow road. The water tank capacity refers to the amount of water a vehicle can carry during an irrigation route. Furthermore, the Key Performance Indicators were identified as the route's total distance and driving duration.

After the contextual analysis, a literature study was conducted to find the best route-planning strategy for the solution model. To find this strategy, it was first necessary to identify the problem that the customers are experiencing. This has been identified to be a Capacitated Vehicle Routing Problem with Heterogeneous Vehicles. Three solution approaches can be used to solve a Vehicle Routing Problem: an exact approach, a heuristic approach and a metaheuristic approach since an exact approach is not able to find solutions for problems with a large number of sensors. The heuristic and metaheuristic approaches were chosen. By analyzing case studies which tackled similar problems to the problem of the customers, two solution approaches were selected:

- The Nearest Neighbour heuristic (NN)
- The Ant Colony Optimization metaheuristic (ACO)

These approaches were combined into the Ant Colony Optimization metaheuristic, which takes in an initial route generated by the Nearest Neighbour heuristic (ACO + NN).

After identifying the solution approaches, the solution model was created in Python. The solution model automatically loads in all the relevant data, calculates the irrigation routes while considering the constraints identified by the contextual analysis, and visualizes the routes so they can be viewed from the dashboard. Furthermore, an Integer Linear Program was developed to test the solution model against the optimal solution.

Experiments were conducted to measure the quality of the routes that the solution model constructs. Here, the routes calculated by the ACO + NN were compared to those constructed by the general Ant Colony Optimization metaheuristic and the optimal solution calculated by an Integer Linear Program. From the experiments, it became evident that ACO + NN outperforms the current solution strategy by an average of 30.01% and 28.84% for minimization on total distance and total driving duration, respectively. The general ACO outperformed the current solution strategy by an average of 31.00% and 29.93% for minimization on total distance and total driving duration, respectively. Furthermore, the ACO + NN had an average gap with the optimal solution of 4.42% and 1.62%. Whereas, the general ACO had an average gap of just 3.72% and 0.74%. Furthermore, the experiments showed that the general ACO produced results 1.07% and 1.19% faster than the ACO + NN. From the experiments has been concluded that the ACO is the better fit for the solution model, providing better results than the ACO + NN.

After the solution model was tested, an implementation and evaluation plan was formed, which could be followed to successfully implement the solution model within the dashboard and evaluate the solution model with customers. The evaluation plan focuses on a beta test with a small number of customers to see if there are any mismatches between the solution model and the actual situations so that these mismatches can be accounted for in the solution model.

Lastly, based on the performed research and conclusions, recommendations are made on how the solution model can be further improved:

- Make it possible for the customers to select the starting location for the routes.
- Store demand of customers
- Set the parameters of the Ant Colony Optimization based on the stored demand
- Create a singular API to retrieve relevant data
- Store pre-calculated distance matrices.

1 INTRODUCTION

This chapter introduces the reader to the research conducted at Connected Green in Hengelo. In Section 1.1, the reader is introduced to the company. Sections 1.2 and 1.3 elaborate on the problem and its identification. Lastly, in Section 1.4, the research design is given.

1.1 Company description

The company at which this bachelor research has been conducted is called Connected Green, a company of Curious Inc. located in Hengelo. Connected Green focuses on giving data insight into green projects, such as the planting of new plants and trees. It does this by selling soil moisture sensors which are planted near the roots of the plants or trees. These sensors send the measured moisture levels to the dashboard of Connected Green, which is in a smart cloud environment. Within this dashboard, all the sensors are linked with the type of plant and soil that needs to be measured by the particular sensor. This is done by using the information of more than 2400 plants and trees given by the plant guide of Griffioen and the Van den Berk catalogue. Using the information of these plants and trees together with the data given by the soil moisture sensors, the dashboard of Connected Green can indicate which plants or trees currently need the most attention in terms of watering. This is important since research has shown that 90% of failing trees and plants can be attributed to either over- or underwatering.

The solution of Connected Green is becoming ever more important as, in recent years, the importance of nature has significantly increased. Plants and trees, apart from improving well-being and biodiversity, are making us more resilient against climate change. As a result, a growing number of green initiatives have been developed, with examples such as the European Green Deal (*Een Europese Green Deal*, n.d.) Nature-inclusive design and the Dutch 'steenbreek'(stone-break) project. (Steenbreek, 2015) It is essential to focus on these green initiatives because nowadays vegetation[1], such as the number of trees and plants, is being reduced due to urbanization. (*Urbanization - Understanding Global Change*, n.d.) To allow vegetation to grow in more urban areas such as city centres, viaducts or rooftops, maintenance and monitoring of the vegetation becomes necessary.

As draughts are happening more frequently and green projects are set up in more difficult or unnatural urban places, the projects are becoming increasingly difficult. As the difficulty increases, it becomes evident that more and more of these green projects might fail. This is problematic as not only is the cost of replacing plants and trees multiple times the cost of the planting materials but the amount of water used for the failed projects is also wasted.

The customers of Connected Green use the soil moisture sensors and the associated dashboard to monitor the green projects and to take necessary action when a plant or tree is withering[2]. The customers, which mostly consist of municipalities, waterboards[3], landscapers, gardeners, provinces, and growers use the solution of Connected Green to plan the routes driven to water the green projects. These routes are often quite costly, so by having insight into which place needs the most attention, the customers are able to save water, reduce CO2 emissions and reduce costs by only watering the plants in need of attention.

[1] plants in general, or plants that are found in a particular area

[2] to make flaccid, shrunken, or dry, as from loss of moisture; cause to lose freshness, bloom, vigour, etc.

[3] regional or national organization which, among other things, concerns itself with nature management in and around bodies of water

1.2 Action problem

Currently, the customers of Connected Green, such as the municipalities, waterboards, etc., are using the sensors and the dashboard provided by Connected Green to plan routes to water the plants or trees in need of hydration. In the case of municipalities with a high number of sensors, it can be necessary to plan multiple irrigation routes in a day. These irrigation routes are mostly done by big tractors with a water tank on a trailer behind it, the water tank contains several thousand litres of water used to irrigate all vegetation in need of attention. Depending on the number of sensors a municipality or other customer has, it can occur that the tractor needs to refill the water tank and start a new irrigation round more than once per day.

These irrigation routes are quite costly and planned by the customers. The customers, often municipalities, have many sensors, sometimes up to 200 per municipality. In general, one sensor measures the moisture levels for around ten trees, meaning there are around 2000 trees that the customers need to monitor and irrigate. As these sensors are spread over a large surface area, e.g., a city, and there is often one central place where the tractors that drive the irrigation routes are stored and refilled with water, the planning of the routes becomes important. This is the case since inefficient routing could result in high driving-related costs. Planning an efficient route is not easy and requires specific skills and knowledge, something the workers at the municipalities are often not trained for. Therefore, the action problem is defined as:

The irrigation routes of Connected Green's customers are not efficiently planned.

1.3 Problem identification

With the action problem defined, it is necessary to now identify the core problem. To do so, a problem cluster has been made around the action problem. The problem cluster is a tool used to display the connections between problems. These problems and relations should be identifiable at a glance (Heerkens, van. Winden, & Tjooitink, 2017). Starting with the action problem, the causes that lead to the problem are identified. As these causes are, in most cases, problems as well, their causes are identified, and this step is repeated until a core problem has been identified. Figure 1.1 shows the problem cluster and the identified core problem.

The action problem relates to customers planning routes by themselves, leading to sub-optimal route planning, which in turn leads to higher costs for the customers. There are three different types of costs first of all, because there is a higher probability of trees dying. As the route is not optimal, the plants or trees that most need attention are not always watered first, leading to a higher probability of the plants and trees drying out. As mentioned in the first section, 90% of failing trees and plants dying can be attributed to either over- or underwatering as the replacement cost of the plants and trees is, as mentioned before, often times multiple times the cost of the planting material. It is necessary that the routes are planned optimally so that there will be a decrease in trees and plants drying out and, thus, a decrease in total replacement costs.

The second cause for the higher costs is that more fuel is used than there would be when driving an optimal route. Since the drivers are driving sub-optimal routes, they drive more rounds than necessary, which leads to an increase in the total distance driven to water all the plants and trees. Logically, this leads to an increase in fuel usage and emissions. Furthermore, as more rounds need to be driven, there is also an increase in labour hours, which also results in an increase in cost for the customers of Connected Green.

Following the problem cluster, it can be seen that all three causes come back to one problem. The planning is done manually by the customers of Connected Green, and since these customers oftentimes lack the knowledge and skill necessary to plan optimal routes, it leads to higher costs for the Customers of Connected Green. From this, the core problem can be identified.
Therefore, the core problem is defined as:

The lack of an automatic route planning solution for the irrigation routes leads to higher costs for the customers of Connected Green.

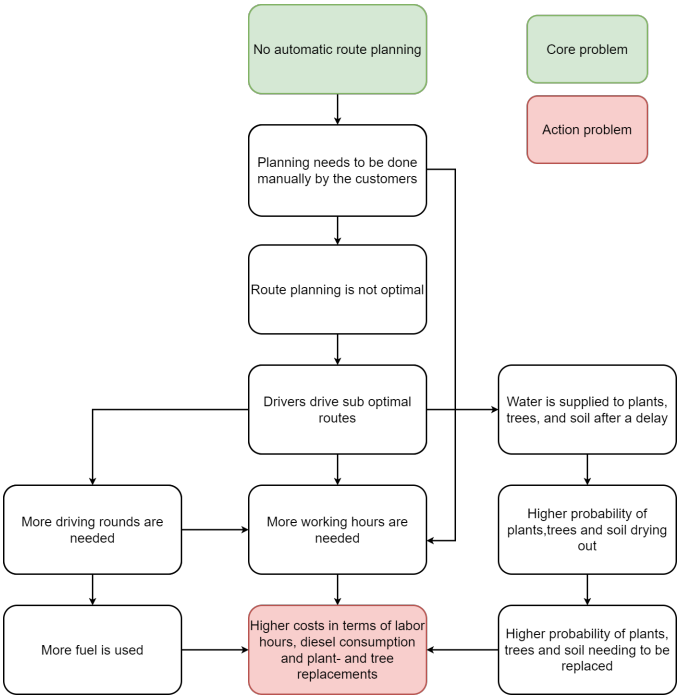

Figure 1.1: Problem Cluster of Connected Green

1.3.1 Gap between norm and reality

As defined in the core problem, the reality is that, at the moment, there is no automatic route planning for the customers of Connected Green. As a result, the irrigation routes planned and driven by the customers are often unnecessarily long and far from optimal. The current routes are longer than they could be in distance driven and driving duration. The long driving distance and duration result in higher costs than what would be necessary due to, among other things, higher fuel consumption and more working hours. These costs can be saved by making a solution that optimizes the route automatically. Therefore, the norm for this research is to create a solution model that will automatically plan and visualize a near-optimal route for the customers of Connected Green based on the relevant data.

The gap is that there currently is no solution to this problem. Therefore, this research aims to create a solution model capable of creating and visualizing the irrigation routes for the customers of Connected Green.

1.3.2 Research questions

The main research question is inherently based on the core problem of Connected Green. The goal of this research is, therefore, to supply Connected Green an automated optimal route planning solution that the customers can use to minimize costs.
The research question is defined as:

What is the most effective approach to develop an optimal route planning model, and how can it be leveraged by Connected Green to minimize the costs incurred by their customers?

The corresponding knowledge questions are:

1. *How are the irrigation routes currently planned by Connected Green their customers, and what are the constraining factors and Key Performance Indicators that influence the route planning?*

   To solve the core problem of Connected Green and supply them with an automated route planning solution, it is necessary first to gain better insight into the current situation. To get this, a context analysis will be performed. The goal of the context analysis is to gather all the information that is needed to plan a route, foremost it is necessary to identify the constraining factors, Key performance indicators (KPIs), and problems that currently arise when planning a route, as these factors will have the biggest impact on the route planning. Furthermore, the current decision-making process will be mapped. In this way, all the requirements, inputs, and outputs connected to the route planning can be identified. To do this, the customers with high numbers of sensors will be contacted, and meetings within the company concerning the problem will be attended.

2. *What are the relevant theories and methods to identify and create the optimal route for Connected Green their customers?*

   After the problem context analysis, a literature study will be conducted to find the appropriate methods which can be used to create the solution. After the relevant literature has been found, it will be analyzed, and the findings will be summarized to find the most relevant method for the solution approach.

3. *Which solution approach can be defined for the optimization of the route planning and what should it look like?*

   This question aims at creating a pseudocode version of the solution approach, which will take all the relationships between the constraining factors, KPIs, data, solving methods, and most fitting heuristic into account. The pseudocode will be checked for completeness with the company and supervisor and can be used as a framework to program the solution. Afterwards, the pseudocode will be programmed in Python which will, using the planning procedure developed, be able to automatically calculate and visualize the irrigation routes for Connected Green their customers.

4. *How can the solution be implemented and evaluated at Connected Green?* The model will be evaluated by testing it against real and made-up routes which the customers could drive to see if there is an improvement. Furthermore, an explanation will be made on how the model can be further evaluated. In order for the implementation of the solution model to be successful, a plan will be developed. This will explain how the solution model can effectively be implemented into the dashboard after the bachelor research has been conducted.

5. *What recommendations and conclusions can be made from conducting the thesis at Connected Green?*

   Based on the outcome of the research at Connected Green recommendations can be formed and conclusions can be made. These will be presented to the stakeholders of Connected Green and the supervisors of the University of Twente.

1.3.3 Scope

The main focus of this thesis is to design a solution model for Connected Green their customers. Through contact with the relevant stakeholders, a general outline of the situation at the customers of Connected Green will be given. This research aims to focus exclusively on producing a solution approach that the Customers of Connected Green can use for the planning of their irrigation routes. Furthermore, this research aims at investigating which methods are best to plan optimal routes in the case of Connected Green their customers. Lastly, this research should provide Connected Green with a solution approach that automatically calculates the optimal irrigation routes for their customers.

1.3.4 Limitations and Restrictions

In its optimality, the solution model will be able to automatically calculate and visualize the optimal irrigation route based on all the constraints. However, given that the time to conduct the research is limited to ten weeks it is necessary to clearly define the limitations of the research beforehand.

Outside of the scope of this research are aspects like a general solution approach or framework for route planning or solution approaches for use-cases other than the use-cases of Connected Green.

Another limitation stems from the fact that customers will need to provide the model with data, e.g., the average amount of water to be given to the trees. In the case that the information given deviates from reality it is possible that the accuracy of the solution also will deviate.

Furthermore, this research mostly has a practical focus. It is a case study for Connected Green and their customers, the findings and outcomes presented may not be directly applicable to other companies or situations. However, in general, it could provide building blocks for further research in studies that have a similar structure or problem context.

To solve both the action and core problem and ultimately answer the research question restrictions are set to help guide the research. The following restrictions are set.

- **Modeling language:** The modelling language chosen to create the optimization and automation solution is Python. As Pyhton has more functionalities, built-in libraries and I have experience with it was chosen as the most suitable language.
- **Information clarity for the driver:** The route information that comes from the model should be readable and interpretable for the drivers of the irrigation routes. They should be able to immediately see what the next stop and the full route is.
- **Data:** The data used for the validation and testing of the model will be of the larger customers of Connected Green. The customers with the smaller number of sensors are not taken into account for this research.

2 CONTEXT ANALYSIS

The objective of this chapter is to create a better understanding of the problem and explain all relevant aspects that will need to be incorporated into the solution model. To accomplish this, Section 2.1 will elaborate on Connected Green and its operations. Section 2.2 will elaborate on the different types of customers Connected Green has and what implications these customers have for the solution model. Section 2.3 introduces the reader to Connected Green's dashboard and its link with moisture sensors and route planning. Section 2.4 describes the way the customers are currently planning the irrigation routes. Section 2.5 covers the identified routing characteristics important for irrigation route planning. Section 2.6 describes the assumptions that have been made. In Section 2.7, the constraints of the problem are discussed. Section 2.8 describes the key performance indicators that indicate a quality route. Section 2.9 elaborates on the algorithm of one of the current route planning strategies, which will be used to test the solution model. Lastly, Section 2.10 summarizes the chapter and answers the research question of this chapter:

How are the irrigation routes currently planned by Connected Green their customers, and what are the constraining factors and Key Performance Indicators that influence the route planning?

All information on which this chapter is built is collected by interviewing the customers with the highest number of moisture sensors, conversations with employees of Connected Green, and analysis of the data within Connected Green their dashboard.

2.1 Connected Green

Connected Green is a company which focuses on giving data insight into green projects, such as the planting of new plants and trees. It does this by selling soil moisture sensors which are planted near the roots of plants or trees. These sensors send the measured moisture levels to the dashboard of Connected Green. Within this dashboard, all the sensors are linked with the type of plant and soil that needs to be measured by the particular sensor. Using the information on these plants and trees together with the data given by the soil moisture sensors, the dashboard of Connected Green can indicate which plants or trees currently need the most attention in terms of watering. Customers use this information to plan irrigation routes to water the plants and trees at these sensors. However, the routes that the customers plan are not efficient.

To facilitate the customers in the irrigation route planning and create a bigger market share, Connected Green wants to create a solution model which can automatically calculate and visualize irrigation routes. The solution model should be able to be integrated into the dashboard of Connected Green.

2.2 The customers of Connected Green

The customers of Connected Green primarily consist of municipalities and landscaping companies engaged in green space maintenance for both individual customers and municipalities. Their usage of the sensors and the dashboard differs greatly per customer. In total, three main use types can be identified.

1. Personal sensors

2. Small number of sensors
3. High number of sensors

2.2.1 Personal

The first type of customers comprises of individuals who use a small number of sensors for personal purposes, primarily in their own gardens. These customers often have less than five sensors. Given the nature of their usage, the development of a solution model for the planning of irrigation routes is not applicable nor beneficial to this type of customer. Therefore, this type of use case is not included in this research.

2.2.2 Small number of sensors

The second type of customers comprises businesses that utilize a relatively small number of sensors (≤ 10) for their operations. Despite the limited sensor count, these customers still encounter the need to plan efficient irrigation routes to optimize their irrigation processes. In the case of several customers, the sensors are spread over a large area such as a city. In these cases, the geographical distances between sensors can be quite large and it can be valuable to have a solution model which is able to plan efficient irrigation routes.

2.2.3 High number of sensors

The third and last type of customers are the dependent customers. These dependent customers, primarily municipalities and landscaping companies managing the green space projects utilize the sensors for business purposes and typically have a substantial number of sensors spread across multiple projects, with each project containing ten or more sensors. A major challenge faced by the customers is the significant time and cost required for irrigation routes due to the widespread distribution of sensors over large areas. Therefore, it becomes necessary to create efficient routes which minimize both cost and time. After talks with customers, several ways of currently planning the routes have been identified. First of all, some customers divide the routes into several different project files with each their own dashboard to create a single route per project. For instance, the municipality of the Hague employs an approach where they divide the routes into separate project files, each having its own dashboard. Afterwards, they use this route to see which sensors currently are in need of the most attention. The irrigation routes start at the most northern sensors and end at the most southern sensor, ultimately driving the routes ”top to bottom”. Secondly, some customers use the sensors to see if the trees or plants need attention and use this indication as a start signal to drive their irrigation routes. Visiting all the sensors and surrounding trees according to a fixed route which is mapped in Google Maps. Both these solutions are not as efficient as they could be but provide an easy fix to the complex problem they are facing.

2.3 Connected Green their dashboard

The customers of Connected Green are able to access the dashboard via app.connectedgreen.nl. Within it, all relevant information for the planning of irrigation routes can be found. This section aims to further elaborate on the three main parts of the dashboard and give insight into how the customers use the information found in these parts to plan irrigation routes.

1. The project overview
2. The sensor overview
3. The map

2.3.1 Project overview

To maintain organization and facilitate efficient planning of irrigation routes, the customers have the ability to create projects within their accounts. As the customers often have several greenery projects in different cities or are working for their own customers it is convenient and necessary to be able to view the sensors of each project individually. The project overview allows for focused management and easy access to project-specific information. Figure 2.1 shows the project overview within the dashboard, here all the projects of the specific customer can be found. In this case, the different in-house projects of Connected Green can be seen.

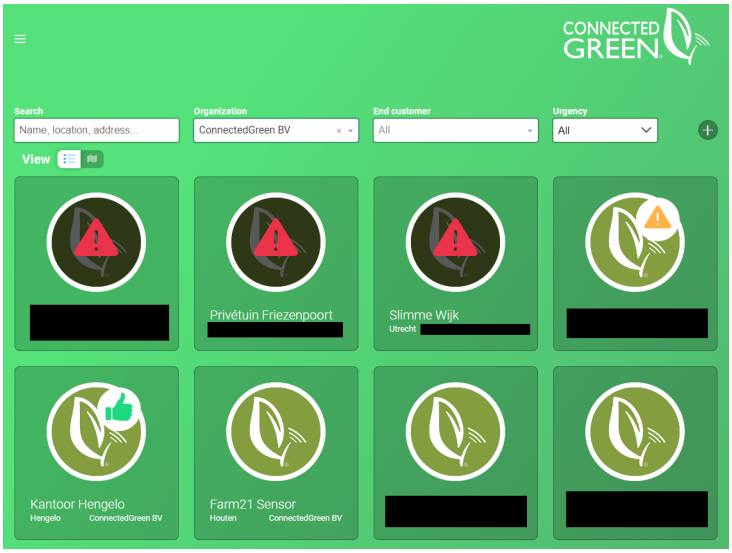

Figure 2.1: Overview of Connected Green their different projects

2.3.2 Sensor overview

Within the project tabs, an overview of all the corresponding sensors can be found. Figure 2.2 displays a sensor overview of customer X. It lists the sensors with their names, connection status, moisture level (%), moisture level thresholds (%), soil type, and plant/tree species. The moisture level thresholds, which can be seen in Figure 2.2 under the status of the sensors, depend on the plant or tree species in combination with the soil type. For each sensor, it is possible to give them their own name. In most cases, this means that the sensors either get the name of the location they are placed on or the name of the plant/tree species the sensors are placed at. The connection status indicates the link between the sensor and the dashboard, the sensors have a typical battery life of 3 to 5 years before they stop functioning. The moisture level at the sensor, expressed as a percentage, is the most important information of the dashboard. The status of the sensor is linked to the moisture level of the sensor. The sensors can have the following statuses which all have corresponding colours.

- Too dry (red)
- Dry (yellow)
- OK (green)
- Wet (yellow)
- Too wet (red)

If a sensor is too dry, immediate action is required to prevent damage to the surrounding plants and trees and keep the plants and trees healthy. Furthermore, if the sensor has the status dry or

wet, it is not necessary to take immediate action. However, it is advised to keep close attention as it is possible that the status will shift to too dry. If a sensor has the status OK, no action is required, and the soil moisture levels are ideal. The status of the sensor shifts to green, yellow or red when crossing a threshold. These thresholds depend on the soil type and plant/tree species and are given in percentages.

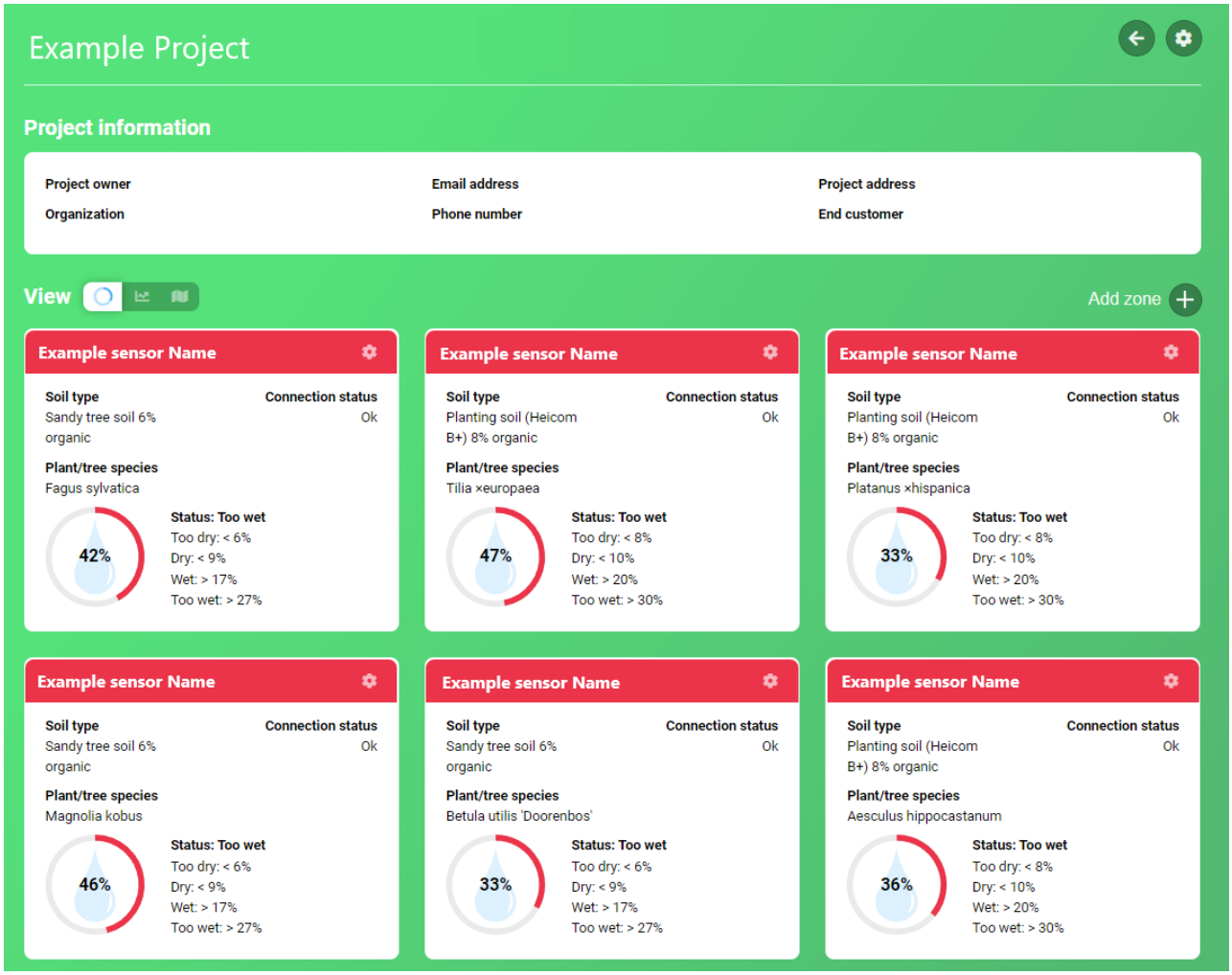

Figure 2.2: Sensor overview of Customer X, in which all sensors have the status too wet.

2.3.3 Map

To ensure effective irrigation route planning, it is crucial to have accurate knowledge of sensor locations. Connected Green addresses this by providing a map feature within its dashboard; here, all the sensors are listed along with their corresponding colour codes. The map provides a comprehensive overview and is particularly useful for intuitively planning irrigation routes, especially when dealing with a small number of sensors. Figure 2.3 illustrates the the map feature, which provides customers with an overview of sensor locations. Within this visual representation, the user can click on individual sensors for quick access to their name, status, and moisture level thresholds.

Figure 2.3: Map of the city of the Hague with its sensors

2.4 Current route planning

After discussions with the largest customers, several route planning strategies have been identified. It is evident that these strategies vary widely and could be improved. The typical process has two steps.

2.4.1 Sensor Identification

The initial step involves identifying which sensors are in need of irrigation. This is done by looking at the dashboard and making a list of all the sensors with the status ”dry” or ”too dry”. The list represents all the to-be-visited nodes.

2.4.2 Route Mapping

Once the sensors that need irrigation are identified, customers plan the irrigation routes. The planning approach is inconsistent and varies from one customer to another. There are several notable methods that customers currently use for route planning:

- Adhering to a predetermined route for every irrigation cycle, regardless of which sensors need attention. These predetermined routes are often saved in Google Maps.
- Starting the route at the most northern sensor, and by moving south, visit all the sensors which are the nearest to the current node. The route ends at the southernmost sensor.
- Visiting all sensors, even if the to-be-visited sensor does not require irrigation. The driven routes are predetermined routes as well and are often saved in Google Maps.

It can easily be seen that the current strategies used for route planning lead to inefficiencies. For instance, visiting sensors that don’t require irrigation can waste time and resources.

2.5 Routing characteristics and requirements

The current route planning methods have been analysed. Although the methods vary from customer to customer, several essential characteristics have been identified that are significant to the creation and execution of irrigation routes. These and other important characteristics already identified in discussions with Connected Green will be elaborated on further. These

characteristics can be seen as requirements for the irrigation route planning and, thus, the to-be-created solution model.

- Vehicle types
- Sensor information
- Water refill points
- Water tank capacities

2.5.1 Vehicle types and road access

The first characteristic identified is the choice of vehicle type. Connected Green's customers employ different types of vehicles based on their specific needs. For example, in urban city centres, using large tractors with large water tanks on the back is impractical and not permitted due to limitations and regulations set by the law. Instead, customers opt for small cars equipped with smaller water tanks. Conversely, big tractors are preferred in more rural areas, where water-carrying capacity is crucial. However, the use of big tractors presents challenges due to road restrictions, making route planning of efficient routes more complex.

The customers indicated that vehicle types must be considered as a requirement for the solution model. In the discussions with the customers, two vehicle types used by the customers have been identified. These are tractors and cars. Therefore, the solution model should be able to calculate routes for the following situations:

- Irrigation using tractors
- Irrigation using cars
- Irrigation using a combination of cars and tractors

The solution model should also consider these vehicles' road accessibilities. The customers indicated that the tractors used for irrigation routes are not allowed on all roads. For example, in the Netherlands, tractors are not allowed to drive on the highway ("Rijksoverheid", "2023"). Therefore, if the solution model calculates an irrigation route that only uses tractors, it should omit highways.

2.5.2 Sensor information

The second characteristic identified is the sensors. All customers have identified the sensors as the main indicator for route planning and execution. In the case of route planning, geolocation is used to create a stopping point. Afterwards, the customers look at the sensor status to see if the surrounding area requires irrigation. If so, the route is planned and driven.

Apart from being identified as the main indicator, the sensor information has also been identified as the most important requirement for the solution model. The most important sensor information consists of the following two types:

- Geolocation
- Moisture level

2.5.2.1 Geolocation

The most important requirement of route planning is to know the locations on which the route is based. Therefore, the geolocational data of the sensors is of utmost importance in the route planning process, as without knowing the locations that need to be visited, it is impossible to plan a route.

2.5.2.2 Moisture level

The moisture level of the sensor indicates if the surrounding area needs irrigation. Recall from Section 2.3.2 that the sensor relays the moisture level information to the dashboard, where the sensor's status changes with the moisture level. Using this information, the sensors that currently need to be irrigated can be identified. Because of this reason, the customers identified that the solution model should automatically add the sensors that need irrigation in the route planning.

2.5.3 Water refill point

The fourth characteristic is the water refill point. This is something which differed greatly between customers. In some cases, the refill points would be along the driven route as surface water out of ditches, canals, etc., would be used. However, this is not allowed in certain parts of the Netherlands (NVWA, n.d.), which forces the customers operating in these parts to fill the water tanks at a central point using tap water.

Therefore, the customers indicated that the solution model should take two situations into account:

- Routes refilling at a central point
- Routes refilling along the route.

During the discussions with the customers, it became apparent that both these situations should be possible in the solution model. If the customers do not refill along the irrigation route, the water tanks of the used vehicles need to be refilled at a fixed central point. In the second situation, the water tanks are refilled along the route. However, the refill points differ per irrigation route, so there is no fixed point at which the tanks are refilled. Therefore, the solution model should be able to calculate routes that do not consider the water tank capacity and visit all the sensors in a single route.

2.5.4 Watertank capacities

The last characteristic is the water tank capacity, which is directly influenced by the choice of vehicles used for the irrigation routes. As discussed earlier, the utilized vehicles can range from small cars to large tractors, and their water tank capacity differs accordingly. Considering the water tank capacity becomes crucial when planning routes as it can directly impact the number of stops that can be made. The customers indicated this as an important requirement for the solution model and that the capacities of the cars are often 1100 litres, and the capacity of the tractors can be upward of 7000 litres.

2.6 Assumptions

Several assumptions have been made to ensure the solution model can be applied to as many customer situations as possible.

2.6.1 Number of trees or flower boxes per sensor

The first assumption concerns the number of trees or flower boxes per sensor. Per sensor, the number of trees or flower boxed can differ. This makes it hard to pinpoint the specific area demand at each sensor as there is no data about the exact number of trees per sensor. It is assumed that the number of trees can range from one to twenty-five per sensor, and the flower boxes can range from 3 to 50. This assumption is based on discussions with customers of Connected Green.

2.6.2 Water demand per sensor

The amount of water a plant or tree needs depends, among other things, on the species, the type of soil it has been planted in, weather conditions and the size of the tree. Due to this, an exact amount cannot be given.(van de Berk, n.d.) However, the customers indicated that the amount of water they give trees ranges from 100 to 150 litres per tree and to the flower boxes is on average, 20 litres. So in combination with the assumption of the number of trees per sensor and the number of flower boxes per sensor, it is assumed that the water demand per sensor can range from 60 litres per sensor to 3750 litres per sensor. However, since the customers often drive the irrigation routes with a single vehicle with a capacity of up to 7000 litres in one day, it is assumed that it is more likely that the water demand per sensor is in the lower ranges since, otherwise, the capacity would already be met while visiting one or two sensors. In the case that only cars are used, which capacity is often times at a maximum of 1100 litres, it is assumed to range from 60 to 1000 litres.

2.6.3 Start- and endpoint

The assumed starting and ending points for the irrigation routes have been set to the project address, which is listed in the sensor overview. This assumption is grounded in practicality; in most instances, the project address is the actual beginning and ending point for irrigation routes.

2.6.4 Service time per sensor

The service time per sensor is assumed to be zero. This is the case because each customer their service times per sensor differ greatly. For example, customer A uses an automated watering system connected to the vehicle used for watering. Their service times are short as they only need to turn on the system and drive past the area that needs irrigation. However, customer B does not use an automated watering system and irrigates the area manually; furthermore, customer B also performs maintenance on the trees or plants at the sensor, increasing the service time even more. Therefore, to be able to provide a solution model for all the customers of Connected Green, the service times are assumed to be zero.

2.6.5 Vehicle types

The vehicle types used within the solution model are assumed to be either tractors, cars or a combination of both. This is because these vehicles are the only ones customers have indicated they are using for irrigation routes.

2.6.6 Car specific routes

It assumed in projects where only the cars are used as vehicles the demand of the sensors cannot be more than the capacity of the car. Therefore, demand per sensor for car-only routes ranges from 60 to 1100 litres. As the customers using the cars indicated that the capacities of the cars are 1100 litres

2.7 Constraints

To be able to model the planning of the irrigation routes effectively, it is necessary to set constraints. These constraints function as boundaries of the problem and will play an important role in creating the solution model, which is discussed in Chapter 4. The following constraints have been set and identified.

2.7.1 Road accessibility

The vehicle types, cars and tractors have different road accessibility. This accessibility is an important constraint for route planning since not all vehicles are legally allowed to travel the same routes. The tractor illustrates this problem distinctly. Due to its limited road accessibility, it frequently takes longer or alternative routes to reach the same destination as a car would. For instance, while cars can easily navigate motorways, tractors are prohibited, forcing them to find alternative routes, often resulting in longer travel times.

2.7.2 Water tank capacity

The water tank capacity directly influences route planning. The water tank capacity depends on the choice of vehicles used for the irrigation routes and essentially dictates how many sensors can be visited per route. A water tank of 1000 litres can water fewer sensors than a tank of 5000 litres, assuming the demand for the sensors is the same.

2.8 Key Performance Indicators

Key Performance Indicators (KPIs) are benchmarks to gauge the effectiveness and efficiency of specific processes or projects. In the case of this solution model, understanding these KPIs can determine the effectiveness of the proposed routing methods. This section elucidates three pivotal KPIs that give insights into the efficiency of the calculated irrigation routes.

2.8.1 Total distance

The first KPI, total distance, encapsulates the overall length of the chosen route in meters. It is a straightforward metric that provides an immediate understanding of the efficiency of the route. A shorter distance, in general, suggests potentially lower costs in terms of fuel and time.

2.8.2 Total driving time

The total driving time, given in minutes, is the second KPI. Stakeholders can assess its efficiency by gauging how long it takes to traverse the proposed route. Similar to the total distance, a shorter driving time often translates to lower operational costs and improved utilization of resources.

2.9 Moving ”South” route planning strategy

To assert the effectiveness of the solution model, it needs to be tested against a current route planning strategy. One of the identified strategies that is currently being used is to start the route at the northernmost sensor and then, by moving south, visit the sensor closest to the current sensor. Continuing this process until all sensors have been visited. This route planning strategy works in the following way:

1. All different locations are ordered based on their latitude.
2. A vehicle starts moving and visits the nodes based on this order.

3. If the node's demand is too high for the vehicle capacity, the vehicle returns to the depot.
4. A new vehicle is selected based on the remaining demand. If the demand is higher than the capacity of the vehicle with the smallest capacity, the vehicle with the larger capacity is selected. If the demand is not higher, the vehicle with the smallest capacity is selected.
5. These steps are repeated until all nodes are visited and a route is created.

This strategy has been converted to an algorithm so that it can be used to test the solution model against the current situation. The algorithm can be seen in Algorithm 1. It takes the demands of the sensors, the latitudes of the sensors, the capacities of the available vehicles, and the vehicle-dependent distance matrices as input and creates an output in the form of a data frame containing both the total distance and order in which the sensors are visited.

Algorithm 1 Moving_South()

Input: demand, locations, latitude, capacities, distance matrices **Output:** data frame with total distance and route orders.

```
Start:
Order the locations based on latitude.
while Not all nodes are visited do
    for each node in the ordered list do
        if the vehicle can supply all demand to the node then
            Add node to the route
            Delete node out of order list
        else
            Return to the depot and select the next vehicle
        end if
    end for
end while
```

2.10 Conclusion

The context analysis provides a comprehensive understanding of the problem. It elaborates on how Connected Green and their customers operate, the different customer groups that exist, and what these customers are currently using the solution of Connected Green for. Furthermore, the dashboard of Connected Green is elaborated on, explaining its functionalities and outlining its link with irrigation route planning. The current route planning of the customers is explained, and it is shown that the current route planning is inefficient. Furthermore, the route planning is analysed, and its main characteristics are identified and explained. The customers' requirements of the solution model are identified and elaborated on. Based on the characteristics, requirements, and discussions with customers, several assumptions about the solution model have been made and elaborated. Based on the assumptions, several constraints are set for the solution model. The constraints and assumptions will serve as input and parameters to create the solution model and guide the literature review. Lastly, one of the current route planning strategies is elaborated on further and converted to an algorithm, which will be used to test the final solution model.

Furthermore, the research question of this chapter has been answered:

How are the irrigation routes currently planned by Connected Green their customers, and what are the constraining factors and Key Performance Indicators that influence the route planning?

Current route planning:

The route planning of the customers differs per customer. However, the main steps include:

1. Identification of the sensors in need of irrigation: using the dashboard, a list of all the sensors which either have the status dry or too dry is made. These are the sensors that will need to be visited.

2. Route planning strategy: depending on the customer, the route is determined differently. It ranges from predetermined routes that visit all the sensors regardless of the moisture level to starting the route at the most northern sensor and driving south while visiting all the nearest sensors in need of attention.

Furthermore, one of the route planning strategies applied by the customers has been described. This strategy functions by starting at the northernmost sensors and by moving to the south, visiting the sensors that are closest to it. This strategy is called the ”Moving South” strategy.

Constraining factors:

The constraining factors that influence route planning are:

1. Road accessibility: the customers can have two types of vehicles: cars and tractors. Each type has different road accessibility, which greatly influences route planning, as in the case of the tractor, it is not allowed on all the roads that the car is on. Therefore, it often needs to take different routes and drive further distances to be able to visit the same sensors.

2. Water tank capacity: this constraint depends on the choice of vehicle and directly influences route planning. For instance, a tank of 5000 litres can water more sensors than a tank of 1100 litres could. (Assuming demand stays the same)

Key Performance Indicator:

The identified Key Performance Indicators are:

1. Total distance: the total distance of a route in meters is an important indicator of the quality of the route. A shorter distance suggests a better route.

2. Total driving time: total driving time in minutes is an important indicator of the quality of a route. A shorter duration is preferred.

3 THEORETICAL FRAMEWORK

This chapter aims to cover relevant existing literature on Vehicle Routing Problems and their solving methods. In Section 3.1 the reader is first introduced to the Traveling Salesman Problem, afterwards, the reader is introduced to Vehicle Routing Problems, the different characteristics of a Vehicle Routing Problem, and the Vehicle Routing Problem of Connected Green is identified. In Section 3.2, the different solving methods for a Vehicle Routing Problem are discussed. Section 3.3 elaborates on the tools used to solve a Vehicle Routing Problem. In Section 3.4, different case studies and the corresponding solving methods, comparable to the situation of Connected Green their customers are presented. Lastly, Section 3.5 summarizes the chapter and provides an answer to the research question of this chapter:

What are the relevant theories and methods to identify and create the optimal route for Connected Green their customers?

3.1 The Traveling Salesman Problem and Vehicle Routing Problem

As the Vehicle Routing Problem (VRP) is derived from the Traveling Salesman Problem (TSP), it is necessary first to explain the Traveling Salesman Problem to develop a good understanding of what the Vehicle Routing Problem entails.

3.1.1 The Traveling Salesman Problem

(Hoffman, Padberg, & Rinaldi, 2001) states that a Traveling Salesman Problem can be stated as: ”if a travelling salesman wishes to visit exactly once each of a list of *m* cities where the cost of travelling from city *i* to city *j* is c_{ij} and then return to the home city, what is the least costly route the travelling salesman can take?” In simpler words, this means that the Traveling Salesman Problem tries to find a route starting at point ”a” through all the cities of a list and back to ”a” while incurring the least cost possible. Below an example of a Traveling Salesman Problem solution can be seen (Kitjacharoenchai et al., 2019).

In relation to the problem of Connected Green, it can, in some customer cases, be seen as a Travelling Salesman Problem. This is because some of the customers refill the water tanks along the route, which can be seen as having unlimited capacity; thus, the demand at the sensors does not influence the irrigation route creation, and since the only other constraint is to have the least distance possible, the problem can be seen as a Travelling Salesman Problem.

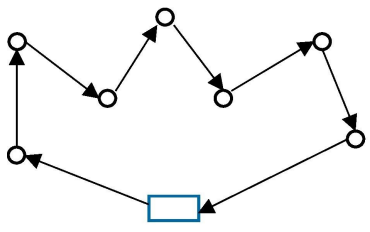

Figure 3.1: Example of a TSP solution

3.1.2 The Vehicle Routing Problem

The Vehicle Routing Problem was first introduced by (Dantzig & Ramser, 1959) in 1959 as the ”Truck Dispatching Problem”. This problem was concerned with the modelling of a route for a fleet of homogeneous trucks which needed to serve the demand for oil of several gas stations starting at a centralized hub. The problem was later generalized by (Clarke & Wright, 1964) into a linear optimization problem which was ”concerned with the optimum routing of a fleet of trucks of varying capacities used for delivery from a central depot to a large number of delivery points.” In other words, the Vehicle Routing Problem concerns itself with designing an optimal routing for delivery from one or more starting points to multiple customers while adhering to certain constraining conditions and operational rules (Zhang, Ge, Yang, & Tong, 2021).

In practice, there exist multiple variants of the Vehicle Routing Problem. This is because of the diversity in constraining conditions and operating rules (Laporte, 2009). Therefore, the Vehicle Routing Problem can be seen as a class of problems. (Lahyani, Khemakhem, & Semet, 2015) provides a taxonomy of the different types of Vehicle Routing Problems. The overview can be seen in Figure 3.2.

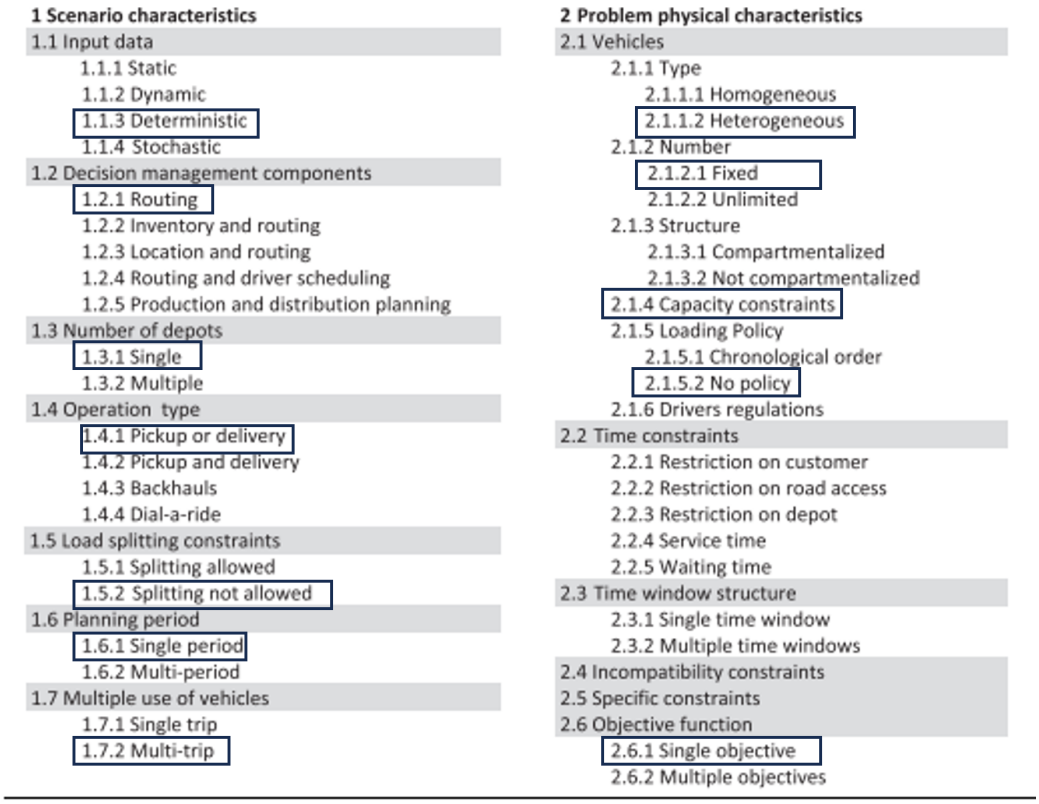

Figure 3.2: Taxonomy of the different types of Vehicle Routing Problems. The highlighted characteristics are relevant for this research.

3.1.3 Identification of Connected Green their Vehicle Routing Problem

The specified characteristics in Figure 3.2 provide insights into the Vehicle Routing Problem (VRP) faced by Connected Green their customers. These characteristics are:

- **Deterministic Input Data:** All necessary data is available and known in advance.
- **Routing Objective:** The primary goal is to devise an near-optimal irrigation route.

- **Single Depot:** Routes are designed with a unified starting and ending point, typically the project address.
- **Delivery Only:** Vehicles are solely responsible for water delivery to the sensors.
- **No Split Deliveries:** A single vehicle must address the demand of a single sensor without splitting.
- **Single Planning Period:** Routing is planned for a singular duration, which is the day of planning.
- **Multi-trip Vehicles:** Vehicles are capable of undertaking multiple irrigation routes within the planning period.
- **Heterogeneous Fixed Fleet with Diverse Capacities:** The fleet size remains constant, but vehicles can differ in type and capacity.
- **Absence of Loading Policy:** There are no specific loading policies.
- **Unified Objective Function:** The primary target is to minimize the total distance covered or the time spent driving.

The Vehicle Routing Problem, which aligns with these characteristics, is the Multi-trip Capacitated Vehicle Routing Problem with Heterogeneous Fleet (MTCVRP). This is a complex extension of the Capacitated Vehicle Routing Problem. However, to reduce complexity, the MTVCRP can be interpreted as a Capacitated Vehicle Routing Problem with a Heterogeneous Fleet but with an unlimited amount of vehicles. While the MTCVRP permits a single vehicle to cover multiple routes, the simplified variant emulates this behaviour by introducing additional vehicles. Essentially, instead of letting a vehicle handle multiple routes, each multi-trip route is allocated to a distinct vehicle.

3.1.3.1 The Capacitated Vehicle Routing Problem (CVRP)

The Capacitated Vehicle Routing Problem is an extension of the general Vehicle Routing Problem. Essentially, it involves determining the most efficient set of routes for a fleet of vehicles with a known capacity to serve a set of customers with known demand. (Cordeau, Laporte, Savelsbergh, & Vigo, 2007) define the constraints for the problem as follows:

1. ”Each customer is visited exactly once by one route.” This constraint ensures that each customer is served by exactly one vehicle route and no customer is left unattended or visited multiple times.
2. ”Each route starts and ends at the depot.” This ensures that the vehicles can easily replenish or unload their goods at the depot.
3. ”The total demand of the customers served by a route does not exceed the vehicle capacity.” This constraint ensures that a vehicle does not need to visit a customer multiple times to fully satisfy their demand.
4. ”The length of each route does not exceed a preset limit L.” This constraint ensures that the routes will not be exceedingly long and will keep fuel consumption, time constraints, and operational efficiency in check.

Due to the fact that the Capacitated Vehicle Routing Problem takes the customer demand and vehicle capacity into account, it is more applicable to real-life scenarios and, because of that, widely used.

3.1.3.2 The Capacitated Vehicle Routing Problem with Heterogeneous Fleet (HCVRP)

The Heterogeneous Capacitated Vehicle Routing Problem, as the name suggests, is an extension of the Capacitated Vehicle Routing Problem, with the extension being the heterogeneous fleet. Unlike the normal CVRP, where all the vehicles are uniform, in the HCVRP, the vehicles can differ in capacity, operating cost, speed and range. The diversity stemming from the vehicles adds an extra layer of complexity, as the problem is not just about finding an optimal route, but also an optimal vehicle allocation. The HCVRP is particularly relevant in modern logistics, as it is more in line with real-life situations (?, ?).

3.2 Solution approaches for Vehicle Routing Problems

In essence, a Vehicle Routing Problem is a combinatorial optimization problem. This means that the goal is to select the best arrangement of options from a finite set to come to an optimal value. Following from (Lenstra & Kan, 1981) it can be stated that almost all the vehicle routing problems can be classified as NP-hard. NP-hardness is a classification of the complexity of a problem. The NP-hard problems are categorized as problems for which no algorithm is known which can exactly solve the problem in polynomial-time exactly. In simpler terms, as the size of the problem grows, the time that is required to solve such a problem increases exponentially. This makes it computationally infeasible to solve large instances of the problem in a reasonable amount of time.

However, there are still ways of solving a Vehicle Routing Problem. The two main methods of solving a Vehicle Routing Problem are with an exact algorithm or using heuristic approaches. (Corona-Gutiérrez, Nucamendi-Guillén, & Lalla-Ruiz, 2022)

3.2.1 Exact algorithms

Exact algorithms guarantee finding the optimal solution to a Vehicle Routing Problem, as they systematically explore the given search space and provide a mathematically proven optimal solution. However, as the problem size increases, the problem's computational complexity increases exponentially. Making it very hard to solve Vehicle Routing Problems in a short time span. (Laporte, 2009) underlines this by stating that exact algorithms are only able to solve Vehicle Routing Problems for up to 100 customers in polynomial time.

One of the most well-known exact algorithms is the Branch-and-Bound algorithm. This algorithm operates by continually dividing the set of all possible routes into progressively smaller subsets. A lower bound on the route length is determined for each subset. These lower bounds steer the subdivision of subsequent subsets. The algorithm continues until it identifies a subset containing a single route. If this route is a lower bound less than those of all other subsets, it is recognized as the optimal route. (Little, Murty, Sweeney, & Karel, n.d.)

3.2.2 Heuristics

In real-life application, the solutions need to be determined quickly, and the number of customers often exceed 100. To still be able to come to a near-optimal solution, heuristics are used. Following from (Nicholson, 1971), a heuristic is defined as a procedure "for solving problems by an intuitive approach in which the structure of the problem can be interpreted and exploited intelligently to obtain a reasonable solution". However, heuristics also have a drawback as they are unable to provide information about the quality of the found solution.

Heuristics come in 3 main types: constructive heuristics, improvement heuristics, and meta-heuristics. (Toth & Vigo, 2014)

3.2.2.1 Constructive heuristics

The goal of the constructive heuristic is to provide a quick solution that does not need to be near optimal. This is because this starting solution will be provided to an improvement heuristic. Out of the constructive heuristics, the two well-known constructive heuristics are the Clark and Wright savings algorithm and the Nearest neighbour algorithm. In the paper of (Avdoshin & Beresneva, 2019), the Clark is tested on a CVRP against several other constructive heuristics and provides the best results however in the same paper, it is also mentioned that in some cases, the Nearest Neighbour algorithm outperforms the Clark and Wright savings algorithm.

Following from (Toth & Vigo, 2014), the Clark and Wright saving algorithm begins by first constructing routes from the depot to each customer. Afterwards, the routes of individual customer pairs (i and j) are merged, while taking the capacity constraints into account. The savings that result from merging these routes are calculated by: $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, where c_{i0} = distance (or cost) from customer i to the depot, c_{0j} = distance (or cost) from depot to customer j, and c_{ij} = distance (or cost) from customer i to j. Based on the calculated savings, the customer pair with the maximum savings is selected, and their routes merge into a singular route. The process of merging continues iteratively until all routes are merged, and a single route with the maximum total savings is left.

Following from (Rio & Harahap, 2023), the Nearest Neighbour Algorithm begins by constructing several routes from the depot to each customer. From there, the next customer to visit is the customer closest to the current location. This is done until all customers are visited while considering capacity constraints. If all the customers have been visited or the vehicle cannot supply another customer, the vehicle returns to the depot, and the route is done.

3.2.2.2 Improvement heuristics

There are two different types of improvement heuristics: intra-route heuristics, and inter-route heuristics. According to (Toth & Vigo, 2014) intra-route heuristics can be any improvement heuristic designed for the Traveling Salesman Problem, e.g., λ-opt (2-opt, 3-opt), Lin and Kernighan algorithm. In both cases, λ edges are iteratively removed and replaced by λ other edges to try and improve the route. In the case of the Lin and Kernighan algorithm, the value of λ is modified dynamically throughout the search. Empirical analysis performed by (Johnson & Mcgeoch, 2007) of several Traveling Salesman Problem heuristics, has shown that The Lin and Kernighan algorithm yields the best results. Following from (Toth & Vigo, 2014) they state that: ”In practice, inter-route improvement moves are essential to achieve good results.”. The inter-route improvement heuristics work by swapping customers or edges between different routes to see if the route will be improved, examples of these kinds of heuristics are Swap, 2-opt*, and Relocate. In most cases, the number of exchanged customers remains small (<2) (Toth & Vigo, 2014).

3.2.2.3 Metaheuristics

According to (Cordeau, Gendreau, Laporte, Potvin, & Semet, 2002), metaheuristics, compared to normal heuristics, perform a more thorough search of the solution space, allowing inferior moves that a normal heuristic normally would not allow. Furthermore, they outperform normal heuristics in terms of quality. However, this increase in quality is often made at the expense

of speed and simplicity. Metaheuristics can broadly be classified into two types: local search methods and population-based heuristics (Toth & Vigo, 2014).

Following from (Toth & Vigo, 2014) the first type, the local search methods, ”explore the solution space by moving at each iteration from a solution to another solution in its neighbourhood.” They do this by moving from an initial solution x_1 and each iteration t it moves from the current solution x_t to a different solution x_{t+1} in the neighbourhood $N(x_t)$ (Toth & Vigo, 2014). If the solution of x_{t+1} is better than the current solution, x_{t+1} will become the new current solution. After the comparison and possible reassignment of the current solution, a new iteration will be started. However, local search algorithms are prone to cycling, which means that it gets stuck on local optima. Two frequently used local search algorithms are Simulated Annealing and Tabu Search.

Following from (Toth & Vigo, 2014) the second type, the population-based methods, ”take their inspiration from natural concepts, e.g., the evolution of species and the behaviour of social insects foraging.” To defer from getting stuck on local optima the population-based methods use high-level guidance strategies based on memory structures, such as neural networks, chromosomes, or pheromones (Toth & Vigo, 2014). However, the heuristics use components of the local search methods to search for near-optimal solutions. Some popular population-based search methods include Ant Colony Optimization (ACO) and Genetic Algorithms (GA).

3.3 Tools for solving a Vehicle Routing Problem

The modelling of a Vehicle Routing Problem entails the utilization of mathematical models to represent the specific problem and its associated constraints. For the purpose of this research, the focus will be on software tools and libraries compatible with the programming language Python. Some well-known and widely known software tools are:

- OR-tools
- Gurobi
- PuLP

3.3.1 OR-tools

OR-Tools is an open-source[1] software suite[2] developed by Google to tackle combinatorial optimization problems. The software suite includes a wide range of solvers which include linear programming, mixed-integer programming, and constraint programming solvers. OR-Tools also offers a Python API, making integrating within the solution model easy.

3.3.2 Gurobi

Gurobi is a mathematical optimization solver designed to handle various optimization problems, including linear and mixed-integer linear programming. It claims to have the world’s fastest solver, which can handle various optimization problems. Furthermore, it has a Python API, making integrating with the solution model easy. Gurobi provides free licenses for Academic purposes and has a great technical support team, which consists of a team of PhD-Level Optimization Experts with fast response times, which makes it a great choice for optimization solutions.

[1] Open-source software is free to use, and the original program can be changed by anyone (Dictionary, n.d.).

[2] A collection of software modules with specific functionalities that are integrated with common interface to support different business processes/objectives(Ayanso, 2014).

3.3.3 PuLP

PuLP is a linear programming library in Python in which you can create and solve linear optimization problems. It supports a variety of solvers, like Gurobi, and has its own built-in solver. Since it is Python-centric, it can easily be integrated with the solution model. Furthermore, the library has a big community, making it easy to ask for help.

3.3.4 Tools that will be used

To effectively measure the effectiveness of a route, a linear program will be created to solve the problem to optimality. Because of its speed, versatility and integrability and the fact that it is possible to acquire a free academic license, Gurobi will be used for this task.

3.4 Case Studies

In the literature study, three metaheuristics were found to be applied to case studies that are comparable to the problem of Connected Green their customers. The found metaheuristics are the Genetic Algorithm, Ant Colony Optimization, and the Tabu Search Method

The first of these three algorithms is the Genetic Algorithm (GA). The problem from (Rachmawati, Sihombing, & Sitorus, 2020) uses the Genetic Algorithm to solve the optimization problem. The Vehicle Routing Problem in the article is a multi-depot (capacitated) Vehicle Routing Problem. The article, which is a case study, uses the use case of the watering of a park in Medan Indonesia. The algorithm finds a solution close to optimality. Following from (Collins, 1998) ”A typical GA represents a solution to a problem in terms of its genotypic features i.e. the basic features, or elements, that make up a solution. These features are represented using symbolic strings (referred to as ”chromosomes”). The most common representation for a GA is a binary (i.e. base 2) string in which each number indicates the presence or absence of a specific element. A set of randomly generated strings is typically used as the initial population for a GA. The chromosomes’ genotypes are then evolved through the application of fitness-biased selection operators, and recombination and mutation reproduction operators. The simulated evolutionary process, involving epigenesis, selection, genotypic survival, and mutation, is repeated until an acceptable solution, or set of solutions, to the problem is discovered.”

The second heuristic is Ant Colony Optimization (ACO) which was inspired by the food-foraging behaviour of ant colonies. In (Zare-Reisabadi & Mirmohammadi, 2015) the heuristic is used to try and solve a site-dependent Vehicle Routing Problem. Site dependency means that different types of vehicles need to be used to go to different locations. The article shows that the heuristic has overall better results than the other heuristic that has been used, namely the tabu search method. (Dorigo & Caro, 1999) explain that an ACO metaheuristic essentially simulates a population of ants which collectively solve the optimization problem. Just as is the case with real ants, the pheromone trail secreted by the foraging ants is stored during the search process. These trails are associated with the connections between the nodes of the problem and initially serve as a long-term memory for the search process. The pheromone trails of the ants are updated based on the quality of the solutions found so that, eventually a solution for the VRP can be found.

The third heuristic is the Tabu Search Method. In (Zare-Reisabadi & Mirmohammadi, 2015) this heuristic is tested against the aforementioned Ant Colony Optimization on the same site-dependent Vehicle Routing Problem. The article concludes that the ACO provides better results. However, the tabu search method heuristic is a close second. This heuristic works by using a

local search procedure to find and generate new candidate solutions. It makes use of a short-term memory that tracks ”tabu” moves to prevent it from getting stuck on local optima. These ”tabu” moves are not allowed to be made again until a certain number of iterations has been reached. In this way, it searches for an optimal solution and ensures it does not get stuck on local optima (Fred, 1990).

3.5 Conclusion

This chapter reviews the literature on Vehicle Routing Problems, focusing on the Capacitated Vehicle Routing Problem (CVRP) and the Capacitated Vehicle Routing Problem with Heterogeneous Fleet (HCVRP). It addresses the main solving methods for Vehicle Routing Problems and specific solving methods for both the CVRP and the HCVRP. Furthermore, it elaborates on case studies which relate to the problem of Connected Green their customers and the solving methods used in these studies.

Furthermore, the research question of this chapter has been answered:

What are the relevant theories and methods to identify and create the optimal route for Connected Green their customers?

Relevant theories and methods to identify the optimal route for Connected Green their customer:

To be able to identify the optimal route for Connected Green’s customers effectively, it is necessary to use an exact method. Even though it is not fast, this will always produce an optimal solution. The method identified and chosen to find the optimal solution will be an integer linear program using the Gurobi solver in Python.

Relevant theories and methods to create the optimal route for Connected Green their customer:

There are several methods of creating a solution for a Vehicle Routing Problem. From the methods, heuristics and metaheuristics are particularly well-suited because of their scalability and efficiency. Therefore both the heuristic and metaheuristic will be used in the solution model. The aim of the heuristic is to provide an initial solution, which the metaheuristic can afterwards improve upon. The type of heuristic that is used within the solution model is the Nearest Neighbour algorithm, due to its computational speed. Furthermore, the metaheuristic chosen to improve on the initial solution of the Nearest Neighbour algorithm is the Ant Colony Optimization algorithm. This decision is based on its ability to outperform other metaheuristics like the Tabu Search method and the fact that the ACO does not require an initial population of solutions, unlike the Genetic Algorithm.

4 SOLUTION DESIGN

The aim of this chapter is to explain the design of the solution model. First, Section 4.1 gives a detailed description of the problem. Section 4.2 describes all the requirements for the solution model. Section 4.3 describes all the assumptions of the model. In Section 4.4, the solution methodology of the model is explained. Section 4.5 describes how the solution model imports data and visualizes the solution. Lastly, Section 4.6 concludes this chapter by summarizing it and providing the answer to the research question of this chapter:

Which solution approach can be defined for the optimization of the route planning, and what should it look like?

4.1 Description of the problem

The problem of Connected Green will be modelled as an Integer Linear Program (ILP) according to the Miller-Tucker-Zemlin formulation (Miller, Tucker, & Zemlin, 1960). The following section will first describe the problem Connected Green's customers are experiencing. Afterwards, the parameters, variables, objective function, and constraints of the problem will be described.

4.1.1 Problem description

The problem the customers of Connected Green are facing is creating efficient irrigation routes to consistently irrigate their plants and trees. The plants and trees are monitored via moisture sensors, indicating when the soil is too dry, thus requiring irrigation. Based on the moisture level and the locations of the sensors, the customers can plan irrigation routes. However, these routes are planned inefficiently. To address this, a solution model will be developed to automate and improve route planning. The solution model can minimize total distance and driving time. This model will determine efficient irrigation routes based on sensor locations, moisture levels, road accessibility, water demand of the sensors, vehicle water tank capacity, and vehicle types.

4.1.2 Parameters

The parameters are the input of the solution model, which affects the decision-making process within the model. The parameters are:

- Total set of sensors: $N = \{1, 2, ..., n\}$.
- Set of sensors plus the depot: $V = \{0, ..., N\}$
- Set of arcs connecting different locations: $A = \{(i, j) \mid i, j \in V, i \neq j\}$
- All sensors have a demand which is nonnegative: $q_i \geq 0$, with $i \in N$. Note that $q_0 = 0$ as the depot does not have demand.
- Types of vehicles: $T = \{1, 2\}$ where 1 is the car and 2 is the tractor.
- The fleet of heterogeneous vehicles: $K_t = \{1, 2, ..., K_t\}$ with each a capacity $Q_t > 0$, with $t \in T$.
- Cost for moving from sensor i to sensor j using vehicle k: c_{ijk}, with $i, j \in N$, $i \neq j$ and $k \in K_t$. The costs can be distance or driving time. The costs can vary per vehicle as the vehicle types have different road accessibilities.

4.1.3 Decision variables

The decision variables are used to indicate the choices made within the model and can only have binary values. In total, there are is one decision variable for this model:

$$x_{ijk}: \begin{cases} 1 & \text{if sensor } j \text{ is visited right after sensor } i \text{ by vehicle } k, \text{ where } i,j \in N, i \neq j, \text{and } k \in K_t. \\ 0 & \text{otherwise} \end{cases}$$

4.1.4 Objective function

The goal of the model is to construct the near-optimal route which minimizes the total cost, which in this case can be either total distance or total driving time. Therefore, the model should minimize the objective function:

$$\min \sum_{(i,j)\in A} \sum_{k=1}^{K} c_{ijk} x_{ijk} \tag{4.1}$$

4.1.5 Constraints of the objective function

The variables in the objective function are subjected to the following constraints.

$$\sum_{j\in V, j\neq i} \sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall i \in N \tag{4.2}$$

$$\sum_{i\in V, i\neq j} \sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall j \in N \tag{4.3}$$

$$\sum_{i\in N} q_i \sum_{j\in V, j\neq i} x_{ijk} \leq Q_k \qquad \forall k \tag{4.4}$$

$$\sum_{i\in V, i\neq 0} x_{0ik} = 1 \qquad \forall k \tag{4.5}$$

$$\sum_{i\in V, i\neq 0} x_{i0k} = 1 \qquad \forall k \tag{4.6}$$

$$\sum_{j\in V, j\neq i} x_{ijk} = \sum_{j\in V, j\neq i} x_{jik} \qquad \forall i \in N, \forall k \tag{4.7}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \tag{4.8}$$

Constraints (4.2) and (4.3) ensure that each sensor is visited once by one vehicle. Constraint (4.4) ensures that the total demand that is met by a vehicle never exceeds the vehicle's capacity; this ensure that a vehicle never visits a sensor whose demand it cannot meet. Constraints (4.5) and (4.6) ensure that a vehicle starts and ends its route at the depot. Constraint (4.7) ensures that the number of times a vehicle visits a sensor equals the number of times it exists the same sensor; this ensures continuity so that a vehicle does not vanish after visiting a sensor. Lastly, constraint (4.8) ensures that the decision variable x_{ijk} can only take on binary values.

4.2 Requirements of the solution model

The solution model should adhere to certain requirements to be able to mirror the real-life situations it will be used for as close as possible. This section defines the requirements the model should have.

- Each moisture sensor can only be visited once per vehicle.
- The vehicles are limited to cars, tractors or a combination of both.
- The vehicles start and end at the same location, the location is the project address found in the sensor overview in the dashboard.
- The road accessibilities differ per vehicle type. For instance, a tractor is not allowed on the motorway.
- The capacity of the vehicles depends on the vehicle type.
- Recall from Chapter 2 that if a vehicle's water tank is empty, it can either be refilled at the depot or using surface water found along the route. Recall from Chapter 3 that if the water tank is refilled along the route, the vehicle can be seen as having unlimited capacity. The choice between refilling at the depot or along the route is specified before the routes are calculated.

4.3 Assumptions

This section lists the assumptions underpinning the solution model. These assumptions simplify the problem and facilitate the creation of a more efficient model.

- The water demand per sensor can range from 60 to 3750 litres for routes driven by tractors or a combination of tractors and cars.
- The water demand per sensor can range from 60 to 1100 litres for routes driven only by cars.
- The service time per sensor is zero.
- A vehicle can drive multiple routes in a single day. However, this is modelled as having unlimited vehicles.
- The product delivered by the vehicles is uniform for all the sensors, so each vehicle is allowed to service each sensor.
- Drivers do not form a limitation. So, only the number and type of vehicles are taken into consideration.

4.4 Heuristics

The solution model is intended to find a feasible solution in an acceptable time for the current and future customers of Connected Green. Since an ILP will not be able to do this for larger instances, a heuristic approach will be used. In Chapter 3, it has been decided that the heuristics used within the solution model will be the Nearest Neighbourhood heuristic and the Ant Colony Optimization metaheuristic. The following sections explain the different heuristics and their most important functions.

4.4.1 Nearest Neigbourhood algorithm

The Nearest Neighbourhood algorithm will provide a quick initial solution that the Ant Colony Optimization algorithm can improve. The algorithm, which can be seen in Algorithm 2, takes as input the list of vehicle types (vehicle_types), the list containing the demand of each sensor (demand_list), the dictionary containing the capacities of the vehicles (capacity_dict), and the dictionary containing the distance matrices of each vehicle type (distance_matrix_dict). Apart

from the distance matrices, all inputs are given by the customer. The algorithm then starts by determining the maximum capacity of the used vehicles. If the demand of any of the nodes exceeds this, the function stops. If not, the function continues and initializes all the nodes with their respective demand.

Following this, a list of permutations is made, each reflecting a separate vehicle order combination. For each permutation, the vehicles will be assigned their specific properties, consisting of the capacity and distance matrix, which can differ per vehicle type.

Within the InitialiazeVehicles() function, all vehicles are assigned their respective distance matrices and capacities. Furthermore, the vehicles are added to a list in the same vehicle order as the current permutation; this list is copied several times, and all copies are added to one big list. Resulting in a two-dimensional list which will be used to dynamically select the next vehicle for which to construct a route, if not all sensors have been visited.

After initialization, the Nearest Neighbour Algorithm will start, and the ComputeRoute() function will determine routes for each vehicle. After the ComputeRoute() function has constructed the routes, they will be saved in a data frame. Upon concluding all the permutations, the FindBestRoute() function will extract the shortest route from the data frame and return the best route order (BestInitialRoute), best route distance (BestRouteValue), and best permutation (BestOption).

Algorithm 2 Nearest Neighbour Algorithm

Input: vehicle_types, demand_list, distance_matrix_dict, capacity_dict
Output: BestInitialRoute, BestRouteValue, BestOption

Start:
Determine the maximum vehicle capacity
Validate vehicle capacity
Initialize nodes with their demands
Generate permutations based on all the different vehicles
for each permutation **do**
    Assign vehicle-specific properties
    InitializeVehicles()
    ComputeRoute()
end for
Extract the best route from the data frame using FindBestRoute()
return BestInitialRoute, BestRouteValue, BestOption

4.4.1.1 Function to compute routes

The ComputeRoute() function, seen in Algorithm 3 and Algorithm 4, is called by the nearest Nearest Neighbour Algorithm. The function essentially runs the route construction algorithm of the Nearest Neighbourhood Heuristic described in Chapter 2, where it starts by constructing a route to one of the unvisited sensors and, from that sensor, visits the sensor nearest to the current sensor. The algorithm keeps constructing routes until all sensors are visited. The ComputeRoute() function takes the sensors (nodes) and vehicles initialized by the Nearest Neighbour Algorithm as input for the algorithm.

The first loop in Algorithm 3, which cycles through all the nodes, ensures that each time a new route is constructed and none of the nodes have been visited, the first point visited after the depot will be a unique node. If possible, due to capacity constraints. In other words, the function ensures that each route starts from a different node to consider all possible permutations of depot-to-node sequences.

At the beginning of this loop, all nodes and vehicles are reset to their original values so that a new iteration can begin without the interference of the last iteration. The function then starts a while loop for each node, which runs until all nodes are visited. Then, if possible, the function adds the unique node from the first loop to the current route, marking it as visited and updating the route distance. This ensures the vehicles start the route at a different node in each iteration. If this is not possible due to capacity constraints, this is skipped.

Afterwards, the function starts looking for the nearest node. This is done using the FindShortestDistance() function, which identifies the nearest node the vehicle can serve using its capacity. If no nodes can be visited, the function returns None, and the route for this vehicle is ended. For a valid node, it is added to the route, marked as visited, and the route distance is updated. Once the loop ends, either by visiting all nodes or early termination, the depot is added to the route and marked as visited. Furthermore, after loop termination, the route distance is updated, and the route details are stored. Lastly, the next vehicle is chosen. This is done dynamically using the function select_next_vehicle().

After all nodes are processed, the stored route details are added to a data frame. Upon completion, the data frame is returned to the Nearest Neighbour Algorithm.

Algorithm 3 ComputeRoute (Part 1)

```
    Input: nodes, vehicles
    Output: data frame with best routes
1:  Start:
2:  for each node do
3:      Reset all nodes and vehicles
4:      while There are unvisited nodes do
5:          if there are no visited nodes then
6:              Add the node to the route.
7:              Mark node as visited
8:              Update route distance
9:          end if
```

Algorithm 4 ComputeRoute (Part 2)

```
        while Route is not completed do
            Find nearest unvisited node using FindShortestDistance()
            if No nodes can be added then
                Add depot to route
                Mark depot as visited
                Update route distance
                Exit while loop
            end if
            Add node to the route
            Set node to visited
            Update route distance
        end while
        Store the route details
        Select_next_vehicle()
        Reset route distance
    end while
    if All nodes have been visited then
        Add route details to the data frame
    end if
end for
return data frame
```

4.4.1.2 Function to select the next vehicle

The function select_next_vehicles() dynamically determines the next vehicle to be used for route creation. As can be seen in Algorithm 5, it takes the list of nodes (sensors), vehicles, the current vehicle index (vehicle_index), and the current list index (list_index) as input. It starts by determining the total demand of the nodes that have not been visited. Afterwards, the function starts a loop that aims to select the most appropriate vehicle for route creation based on the current demand and available vehicles. Within the loop, the function first attempts to identify any unused tractors or cars from the current list of vehicles. Afterwards, the following logic determines the next vehicle to be used:

- If there is an available tractor and the total remaining demand exceeds the capacity of a car, the tractor is chosen due to its higher capacity.
- If there is an available car that can at least visit one more node, the car is selected.
- If both are not true but a tractor is still available, the tractor is chosen by default.
- If there is no tractor available, the list index is incremented with 1 to consider the next set of vehicles.

This selection procedure is based on two assumptions. First, if the remaining demand is too high to be met by a car. It is better to use a tractor since it has more capacity and thus can visit more sensors in a single route. Secondly, longer routes, which visit more sensors, are preferred over shorter routes. The shorter routes require more visits to the depot, leading to an increase in travel distance from and to the depot. By driving longer routes, it is possible to minimize the depot visits and the overall traveled distance can be reduced. Furthermore, it also results in fewer vehicles needed to drive the irrigation routes.

Algorithm 5 select_next_vehicle()

Input: vehicles, nodes, list_index, vehicle_index
Output: vehicle_index, list_index

Start:
Determine the total demand of all unvisited nodes
while No new vehicle is selected **do**
    Determine if there is still an unused tractor in the list of the available vehicles
    Determine if there is an unused car in the list of the available vehicles
    Determine the capacity of a car
    if an unused tractor is available, and the remaining demand is higher than the capacity of a car **then**
        return the index of this unused tractor
    else if an unused car is available and there is at least one node that can be added to its route **then**
        return the index of this unused car
    else if There is an unused tractor **then**
        return the index of this unused tractor
    else if there is no unused tractor **then**
        List index+=1
    else if there is no unused car **then**
        List index+=1
    end if
end while

4.4.2 Ant Colony Optimization algorithm

The Ant Colony Optimization algorithm (ACO), which will be used to create near-optimal routes, is a modified version of the general ACO. Whereas the normal ACO does not take in an initial solution, the ACO used in the solution model does. It takes the initial route order found by the Nearest Neighbour algorithm and improves upon it. Following from (Wang, Ma, Li, Zhai, & Qiao, 2020), the general ACO algorithm works in the following way:

1. Initialization:
    - A number of ants are divided over the nodes of the problem.
    - All connecting paths between the nodes have a pheromone concentration, which initially is set to a uniform value or in our case to values based on a heuristic.
2. Ants moving:
    - Ants start moving from one node to another based on pheromone concentration on the path and the attractiveness of the move, which often is distance, cost or time. The probability of an ant k moving from node i to node j is given by:
      $p_{ij}^k = \frac{[\tau_{ij}]^\alpha \times [\eta_{ij}]^\beta}{\sum [\tau_{ij}]^\alpha \times [\eta_{ij}]^\beta}$ where,
        - τ_{ij} is the pheromone concentration from node i to node j.
        - η_{ij} is the attractiveness of the move from node i to node j which is given by: $\eta_{ij} = \frac{1}{d_{ij}}$ with d_{ij} being the distance, time or other cost.
        - α is the parameter which decides the relative importance of the pheromones.
        - β is the parameter which decides the relative importance of the attractiveness.
3. Pheromone Update:

- After all ants have created a path, the pheromone levels on the trails are updated.
- To simulate the natural evaporation of pheromones, a percentage of the pheromones is evaporated from all the paths. This is given by:
  $\tau_{ij} = (1 - \rho) \times \tau_{ij}$ where,
  - ρ is the evaporation rate.
- On the paths the ants have used, pheromones are deposited, with the amount being inversely proportional to the lengths of their paths (or the quality of their solution). This is represented as:
  $\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$ where,
  - Q is a constant, and L_k is the length of the path taken by ant k.
- The new pheromone levels on the paths are then: $\tau_{ij} = \tau_{ij} + \sum_k \Delta\tau_{ij}^{k}$

4. Termination:
   - The algorithm is terminated when a certain condition is met, for instance, a number of iterations.

The ants choose to travel to nodes based on the strength of the pheromone trails left by earlier ants. The Ant Colony Optimization works by dividing all the available ants over the nodes. From there, the ants will choose the next node based on the probability distribution.

4.4.2.1 Ant Colony Optimization algorithm

The main function of the ACO, which can be seen in Algorithms 6 and 7, has the following inputs: the list containing the types of vehicles (vehicle_types), list with the demand of each sensor (demand_list), a dictionary containing the distance matrices of the different vehicles (distance_matrix_dict), and in the case of the modified Ant Colony Optimization Algorithm a list containing the initial route calculated by the Nearest Neighbour Algorithm (initial_route).

The algorithm starts by initializing all the parameters of the ACO. These are:

- Number of ants: in the algorithm, the number of ants is set to the number of nodes divided by two that need to be visited to ensure the ants are divided over the nodes.
- Alpha: this represents the extent of the group cooperation of the ants, meaning that the higher value α attains, the more likely an ant will choose a path based on pheromones.
- Beta: this represents the extent to which an ant will favour taking a path with a lower distance, time or cost.
- Evaporation rate: this must be $0 \geq$ Evaporation rate ≤ 1.
- Initial pheromones: this refers to the pheromones the initial route provided by the Nearest Neighbour algorithm will receive. This is only done for the modified Ant Colony Optimization algorithm.
- Number of iterations
- Q: a constant.

After initializing all the parameters, the parameters, as well as the inputs given to the algorithm are used to initialize the Ant Colony Optimization algorithm.

After initializing, Colony.optimize() is called, starting the optimization algorithm.

Algorithm 6 Modified Ant Colony Optimization Algorithm

Input: vehicle_types, demand_list, distance_matrix_dict, capacity_dict, initial_route
Output: best_route_order, used_vehicles, best_route_loads, best_route_distances

1: **Start:**
2: Initialize the parameters of the ACO.
3: Initialize the Ant Colony with the parameters
4: Colony.optimize()
5: **return** best_route_order, used_vehicles, best_route_loads, best_route_distances

Algorithm 7 General Ant Colony Optimization Algorithm

Input: vehicle_types, demand_list, distance_matrix_dict, capacity_dict
Output: best_route_order, used_vehicles, best_route_loads, best_route_distances

1: **Start:**
2: Initialize the parameters of the ACO.
3: Initialize the Ant Colony with the parameters
4: Colony.optimize()
5: **return** best_route_order, used_vehicles, best_route_loads, best_route_distances

4.4.2.2 The optimization function

The function Colony.optimize() will run the actual ACO algorithm. As can be seen in Algorithms 8 and 9, the function starts by checking if any node's demand exceeds the maximum vehicle capacity. If so, the algorithm is halted. This is because all nodes need to be visited, and all demand of the node needs to be met during a single stop.

In the case that the algorithm is not halted, the iteration is started. The iteration starts with resetting all vehicles and nodes to their initial state. Afterwards, it first increments the counter with one, which lets each vehicle start on a different node. After this is done, it iterates over all vehicles and computes the route for the vehicle using Colony.compute_solution. Throughout the iterations, the method will update the best route if it discovers one that outperforms the current best route.

After iterating over all the vehicles, the Colony.update_pheromone() functionality is called. This evaporates a portion of the pheromones on all paths and adds pheromones to the paths taken by the vehicles. As each iteration ends, the route list maintained by the Colony is cleared, providing a clean slate for the next iteration.

After all iterations are done, the best route, along with its information, is extracted and returned.

Algorithm 8 Colony.optimize() (Part 1)

Input: **Output:** best_route_order, used_vehicles, best_route_loads,

1: **Start:**
2: Verify demand with Colony.check_demand()
3: **if** a node's demand exceeds maximum vehicle capacity **then**
4: Stop algorithm
5: **end if**

Algorithm 9 Colony.optimize() (Part 2)

```
6:  for each iteration do
7:      for each vehicle do
8:          Increment counter with one
9:          Reset all vehicles and nodes to their initial states using Colony.reset()
10:         Compute the best route for current vehicle using Colony.compute_solution()
11:         if the current route is better then
12:             Update best_route and its distance
13:         end if
14:         Clear the current vehicle its route list.
15:     end for
16:     Update pheromone levels using Colony.update_pheromone()
17:     Clear the Colony route list
18: end for
19: Extract best route details: best_route_order, used_vehicles, best_route_loads,
    best_route_distances from best_route
20: return best_route_order, used_vehicles, best_route_loads, best_route_distances from
    best_route
```

4.4.2.3 Function for solution computation

The function Compute.solution() determines near-optimal routes for vehicles while accounting for vehicle capacity and various vehicle types. It can be seen in Algorithm 10 and starts by resetting the total distance to zero and adding the depot to the route. Afterwards, it evaluates whether the node corresponding to the counter index can be added to the route. If possible, the node is added to the route.

The process then enters a loop which keeps running until all nodes have been visited. Within the loop, the probability distribution is computed for visiting each node from the current node. This same function simultaneously discerns if a vehicle has enough capacity to visit anymore nodes. If the capacity is exhausted, the depot is added to the route, and the route is saved using the vehicle.save_route() function. If not, the next node is selected using a random choice based on the probability distribution.

Within the vehicle.save_route() function, the routes are saved, and the vehicle is reset so that it starts from the depot and will be able to visit the unvisited nodes. Furthermore, if there are several vehicle types, the vehicle type will change depending on the counter within each vehicle. Each time a route is saved, this counter gets incremented and assigns the vehicle type with the counter index in the vehicle_types list.
After the vehicle type is assigned, the while loop is continued.

If the capacity is not yet exhausted, the next node that was selected will be added to the route, and the loop continues. Upon visiting all nodes, the depot is added to the last route and saved. The function concludes by summing all the distances travelled by the vehicles. It returns the vehicles, their respective routes and the total distance.

Algorithm 10 Colony.compute_solution()

Input: vehicle, counter **Output:** vehicle routes, total distance

Start:
Initialize total distance to zero
Add depot to route
if node at the counter index can be added to route **then**
    Add node to route
end if
while there are unvisited nodes **do**
    Compute probability distribution for next node using Colony.compute_probabilites()
    Select next node using a random choice based on the probability distribution.
    if force_depot is True **then**
        Add depot to route and save the route using vehicle.save_route()
    else if next node is the depot **then**
        Add depot to route and save using vehicle.save_route()
    else if next node can be added to route **then**
        Add next node to route
    end if
end while
Add depot to route and save using vehicle.save_route()
Calculate total distance travelled
return vehicle routes, total distance

4.4.2.4 Function for computing the probability distribution

The function Colony.compute_probabilities(), which can be seen in Algorithm 11, is used to determine the probability distribution of visiting nodes from the current node. The function takes in the current vehicle and the current node (current_node) and works in the following way:

1. All pheromone values τ_{ij} for the paths that can be taken from the current node are loaded in.
2. All the distances/travel times for the paths that can be taken from the current node are loaded in.
3. All the η_{ij} values are computed using: $\eta_{ij} = \frac{1}{d_{ij}}$ with d_{ij} being the distance, time or other cost.
4. The probability distribution is determined using: $p_{ij}^k = \frac{[\tau_{ij}]^\alpha \times [\eta_{ij}]^\beta}{\sum [\tau_{ij}]^\alpha \times [\eta_{ij}]^\beta}$
5. The vehicle is checked on if it has enough capacity to visit a node, if not, force_depot is set to True
6. The probability distribution together with force_depot is returned to Colony.compute_solution().

Algorithm 11 Colony.compute_probabilities()

Input: vehicle, current_node **Output:** probabilities, force_depot

Start:
Access all the pheromones of the points that can be visited from the current node
Access the distances/traveltime from the points that can be visited from the current node.
Compute the η values
Compute the probability distribution for moving from the current node to the other nodes
if vehicle has too little capacity to visit a node **then**
    force_depot is set to True
end if
return probabilities, force_depot

4.4.2.5 Function for updating pheromone trails

The function Colony.update_pheromones(), which can be seen in Algorithm 12, is used to update the pheromone trails between the nodes. It takes the current vehicle and the current node (current_node) as input and works in the following way:

1. Evaporate a portion of all the pheromones on the paths. Using: $\tau_{ij} = (1 - \rho) \times \tau_{ij}$ where, ρ is the evaporation rate.
2. Add pheromones on the paths traversed by the ants using: $\tau_{ij} = \tau_{ij} + \sum_k \Delta\tau_{ij}^k$ where, $\Delta\tau_{ij}^k = \frac{Q}{L_k}$.

Algorithm 12 Colony.compute_probabilities()

Input: vehicle, current_node **Output:** probabilities, force_depot

Start:
Evaporate a portion of all pheromones trails.
for each vehicle in Colony.route_list **do**
    for each route of the vehicle **do**
        for each node in the route **do**
            Add pheromone on the path between the current node and next node in the route.
        end for
    end for
end for

4.5 Solution model

Before the Nearest Neighbour algorithm and the ACO can determine a near-optimal route, it is first necessary to import and convert all relevant data. After this has been done and the routes have been determined, it still is necessary to create a visual representation of this route so that the customers of Connected Green are able to view it. All relevant data must first be imported to use the Nearest Neighbour and Ant Colony Optimization. This section will elaborate on how all relevant data has been imported, converted and how the final route is mapped.

4.5.1 Loading in all relevant data

The final solution model should be designed to determine the most efficient irrigation route automatically. Almost all required data for this model is available on Connected Green their

dashboard. The most important data for the solution model on the dashboard consists of several variables:

- zoneId: the sensor name used to identify which sensor is which in the routes.
- Longitude and latitude pairs: used to place the sensor on a map and calculate the distances from and to each point.
- Moisture level and thresholds: will be used to indicate if a sensor requires attention and thus will be included as a watering point in the route.

As no dataset is available, the data is either imported into the model using the APIs of Connected Green or imported into the model as user input. The API filters the sensors so that the data frame only includes sensors with the readings dry and too dry. Ultimately the APIs return a data frame which includes all sensors that need attention, their latitude, longitude and zoneId.

The user input covers the non-existent data needed to calculate the routes. The user input data consists of the following:

- ProjectId: the identifier used for Connected Green their API to load the sensors of a specific project.
- Refill_at_depot: Boolean value that represents if a vehicle refills its capacity at the depot or along the route.
- Capacities: the capacities of the vehicles differ per customer.
- Choice of vehicle: this can be a car, tractor or both.
- Number of vehicles: the number of vehicles available of each type.
- Demand of sensors

4.5.2 Graphhopper

To determine a near-optimal route, it is necessary to know the distances and/or driving times from and to each sensor. To accomplish this, Graphhopper is used. Graphhopper is an open-source routing library and is based on OpenStreetMap data. By hosting the Graphhopper engine locally, it is possible to add constraints to the distance calculations based on the type of vehicle used. For example, in the case of tractors, it is possible to exclude motorways and roads with a speed limit of 100 km/h and include private roads normally not considered. In this way, it is possible to create different distance matrices for each vehicle type, closely following the real-life situations the customers of Connected Green face while executing the irrigation routes. In the case of the solution model for the customers of Connected Green, constraints are set for the tractor vehicle type. The constraints are modelled to take into account the legal road accessibility regulations for tractors and the size constraints of the tractor. For example, a tractor with a width of 2.5 meters cannot use roads that are at most 2 meters wide. Furthermore, the Graphhopper API is also used to generate the paths to map the lines connecting the nodes on the map.

4.5.3 Folium

After calculating the final route, it still has to be mapped. This has been done by using folium. Folium is a Python library used for visualizing geospatial data. It takes the final route information, the latitude and longitude, sensor names and the paths generated by the Graphhopper API as input to map all the sensors and connect them in the order dictated by the solution method. After everything has been mapped, the map is available as an interactive .html file and can be viewed by the customers.

4.6 Conclusion

This chapter describes the design and workings of the solution model. It first gives a clear description of the problem. It does this by first summarizing the problem. Thereafter, it describes the problem as an Integer Linear Program with its parameters, decision variables, objective function and constraints. Afterwards, the requirements of the solution model are described, such as road accessibility, refill points, and the start- and end locations. Next, the assumptions of the solution model are described. These assumptions consist of the water demand per sensor, non-existent service time, driving multiple routes, uniformity of the product and driver limitations. After this, the heuristics used within the solution model are elaborated on. The heuristics consist of a Nearest Neighbourhood algorithm and an Ant Colony Optimization algorithm. The logic and integration of these heuristics within the solution model are described. Afterwards, the solution model is described, and the manner in which it loads and converts the relevant data and transforms the calculated route into a visual representation is elaborated on.

Furthermore, this chapter has given an answer to the research question:

Which solution approach can be defined for the optimization of the route planning, and what should it look like?

Solution approach:
The solution approaches that can be defined for the optimization of the route planning is a combination of the use of a Nearest Neighbourhood algorithm and an Ant Colony Optimization algorithm. Where the Nearest Neighbour Algorithm provides a quick initial route, which the Ant Colony Optimization uses as input as an initial pheromone trail. The ACO then calculates the near-optimal route, simulating the food-foraging behaviour of ants.

The way this approach should look is described in Section 4.4. This section elaborates on the logic used within the solution model in detail.

5 EVALUATION AND IMPLEMENTATION

This chapter aims to elaborate on the evaluation and implementation of the solution model at Connected Green. First, Section 5.1 will elaborate on the design of the experiments, data that will be used, scenarios of the experiments, Key Performance Indicators, and the validation of the experiments. Section 5.2 will elaborate on the parameter settings of the ACO. Section 5.3 will elaborate on the solution methods the solution model will be tested against. Section 5.4 elaborates on the results of the experiments as well as the output of the solution model. Section 5.5 elaborates on the evaluation and implementation plan for the solution model at Connected Green. Lastly, 5.6 concludes the chapter by summarizing and providing an answer to the research question of this chapter:

How can the solution be implemented and evaluated at Connected Green?

5.1 Experimental design

In order to evaluate the route-creating performance of the Ant Colony Optimization algorithm used within the solution model, it is necessary to perform several experiments. The first experiments are related to the parameter setting for the Ant Colony Optimization metaheuristic and Ant Colony Optimization with the initial solution metaheuristic. Secondly, the numerical experiments aim to test both Ant Colony Optimization Algorithms against the current route planning method described in Chapter 2, the Nearest Neighbour algorithm and an Integer Linear Program. This section will discuss the dataset on which the experiments will be tested, how the experiments are designed, what the Key Performance Indicators are, and how the experiments can be validated.

5.1.1 Dataset

Unfortunately, not a significant amount of data is available to test the algorithm. In the case of customers, the customers with the largest number of sensors in a certain area are around 20 sensors. To still be able to test the algorithm effectively, a dataset has been created for the city of the Hague. The dataset includes latitude and longitude points for 100 different locations for this city, distance and duration matrices for these points for both vehicle types and artificial demand for the 100 locations.

5.1.1.1 Water demand

There is no available data on the demand of the sensors. However, as mentioned in Chapter 4, the customers provided an average amount of water and amount of trees or flower boxes per sensor. Under the same assumption mentioned in Chapter 4, the water demand per sensor will range differently for the vehicle combinations.

- Tractors: the water demand ranges from 100 to 3750 litres.
- Car: the water demand ranges from 60 to 1000 litres.
- Combination of tractors and cars: the water demand ranges from 60 to 3750 litres.

As is also underlined in Chapter 2, the demand distribution is not uniform and tends to skew to lower values. Therefore, to be able to simulate the demand effectively, a probability distribution

needs to be used, which can model this situation. To be able to do this, the triangular distribution is used. This distribution is used when there is a known relationship between the variable data but when there is relatively little data available. It is also often referred to as a "lack of knowledge" distribution. It is an ideal distribution based on the maximum and minimum values and the most likely outcome: the mode. The probability density function $f(x)$ for a triangular distribution is given by. (Kissell & Poserina, 2017)

$$f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & \text{for } a \leq x < c, \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{for } c \leq x \leq b, \\ 0 & \text{otherwise.} \end{cases}$$

where:

- a is the minimum value,
- b is the maximum value
- c is the mode.

The minimum and maximum values of all vehicle types are known, however, the modes are not. Although the modes themselves are not known, it is given that the water demand tends to be in the lower ranges as mentioned in Chapter 2. Therefore the modes are assumed to be:

- Tractors: 300
- Cars: 100

Using the modes two probability density functions can be created:

1. Tractors:
$$f_1(x) = \begin{cases} \frac{2(x-100)}{(3750-100)(300-100)} & \text{for } 100 \leq x < 300, \\ \frac{2(3750-x)}{(3750-100)(3750-300)} & \text{for } 300 \leq x \leq 3750, \\ 0 & \text{otherwise.} \end{cases}$$
where: $a = 100$ (minimum value), $b = 3750$ (maximum value), and $c = 300$ (mode).
2. Cars:
$$f_2(x) = \begin{cases} \frac{2(x-60)}{(1000-60)(100-60)} & \text{for } 60 \leq x < 100, \\ \frac{2(1000-x)}{(1000-60)(1000-100)} & \text{for } 100 \leq x \leq 1000, \\ 0 & \text{otherwise.} \end{cases}$$
where: $a = 60$ (minimum value), $b = 1000$ (maximum value), and $c = 100$ (mode).

To determine the demand for a combination of cars and tractors, a random number is generated between 0 and 1. In the case that the random number is higher than the value p, the demand gets simulated using the car its probability distribution; otherwise, the distribution of the tractors will be used. In other words:

$$f_3(x) = \begin{cases} f_1(x) & \text{for } p \leq 0.5 \\ f_2(x) & \text{for } p > 0.5 \end{cases}$$

The value of p is set to 0.5 so that the probability of choosing either the demand out of the distribution of the car or the tractor is equal.

5.1.1.2 Latitude and latitude points

The geolocational points are generated using a Python script which takes a random value within a polygon. The polygon covers the entire city of the Hague and excludes points in places where no route can be created, such as points in the sea. The area in which the points can be created can be seen in Figure 5.1.

Figure 5.1: Map of the polygon of the Hague used to create geolocational points made in Google Maps

5.1.2 Scenarios

All experiments will be run using a Python script and will be able to automatically produce a .csv file with the results of the experiments. The experiments will test the four different heuristic approaches against the Mixed Integer Linear program using the dataset created for the city of the Hague. The customers will be n= 10, 25,50,100, and the optimization will be based on total distance and driving duration. In total, there will be three different experiment scenarios:

1. Optimization using cars
2. Optimization using tractors
3. Optimization using a combination of tractors and cars
4. Optimization using a combination of tractors and cars, where tractors visit the sensors outside the city centre and cars the sensors in the city centre

5.1.3 Key Performance Indicators

The Key Performance Indicators of the experiments will be able to indicate how good the performance is of the different heuristics. The Key Performance Indicators will consist of:

- Running time:
  An important Key Performance Indicator of the performance of a heuristic is the total time it takes the heuristic to produce its result. In the case of Connected Green, it is not desirable to use a heuristic that takes a significant time to calculate a solution, as the heuristic will need to be incorporated in the dashboard.
- Total distance:
  An important indicator for the solution quality found by the heuristics is the total distance in meters (in the case that the optimization is based on distance). A lower distance is preferable and indicates a good quality solution.
- Total driving duration:
  Just as the total distance, the total duration is an important indicator. It is given in seconds and will only be provided if the optimization is based on driving duration.
- Difference with the ILP:
  The ILP, just like the heuristics, provides the solution with its total distance or driving duration (dependent on which optimization). As an ILP determines the optimal solution, the difference in percentage between the solution of the heuristics and ILP is also an important indicator of a found solution.
- Difference with the current heuristic:
  As the current route planning strategy of the Hague has been converted to a heuristic, it is possible to compare the Nearest Neighbour algorithm and the Ant Colony Optimization algorithm with this strategy. This indicates the difference in percentages when the proposed algorithms are used instead of the current strategy.

5.1.4 Validation of the experiments

This section will cover the computational setting and the randomness of the ACO.

- Computational setting:
  All experiments are run on the same computer, a Windows computer with 16 GB of RAM, and an Intel i7-9750H core and 2.60GHz core are used. All code is written in Python 3.10.12 using Visual Studio Code 1.81.1. Furthermore, the ILP is also coded in Python using the package and solver of Guribo. The Gurobi solver is free to use under an academic license.
- Randomness of the ACO:
  Since the ACO uses probability to model the foraging behaviour of the Ant, the constructed solutions can differ per time it is running. To account for this randomness, all experiments are run five times.

5.2 ACO tuning

Before the experiments can be run, it is necessary to set the different parameters of both ACOs. In Chapter 4, the parameters of the ACO have been discussed. The algorithm will provide different quality solutions based on these parameters. This section elaborates on the different parameters and their values.

- Number of ants:
  The number of ants is set to be the number of nodes to be visited divided by two, to ensure that computational time is low.
- Alpha:
  Alpha is the parameter that determines the extent of the group cooperation of the ants, meaning that the higher value α attains, the more likely an ant will choose a path based on pheromones.
- Beta:
  Beta represents the extent to which an ant will favour taking a path with a lower distance, time or cost.
- Evaporation rate: The evaporation rate determines how much of the pheromones on each path dissipate. This value has been set to 0.5.
- Initial pheromones (only for the ACO with initial route):
  This parameter determines the amount of pheromones already on the initial route. With the initial round found by the Nearest Neighbour algorithm. As the pheromones on all paths are set to 1, the initial route has been set to 1,5 to ensure that the ants are more likely to follow the initial route and then deviate from it.
- Number of iterations: this is set to 100 to reduce computational times.
- Q: is a constant. It determines the amount of pheromones that will be added to each path if it has been taken. Recall from Chapter 4: $\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$. In the experiments, Q will attain the lowest travel distance from a node to another node. This way, the maximum a pheromone will be updated by is one, and the higher the distance, the lower the added pheromone will be.

Alpha and beta have been selected based on the lowest average attained value of five experiments with n = 25 for beta and alpha in the range [0.5,5]]. In these experiments, every possible alpha-beta pair has been used to compute a solution for the total distance five times. After all experiments, the alpha-beta pair with the lowest average total distance was selected.

5.2.1 ACO and Nearest Neighbour combination

For the ACO with the initial route, the alpha and beta values that reached the lowest average are: $\alpha = 0.5, \beta = 1.5$. A low alpha value and a relatively high beta value indicate that the ants will favour taking paths of lower distance instead of exploring pats that at first may have larger distances. The graph with all the different alpha-beta pairs and their average distances for the ACO with the initial route can be seen in Figure 5.2.

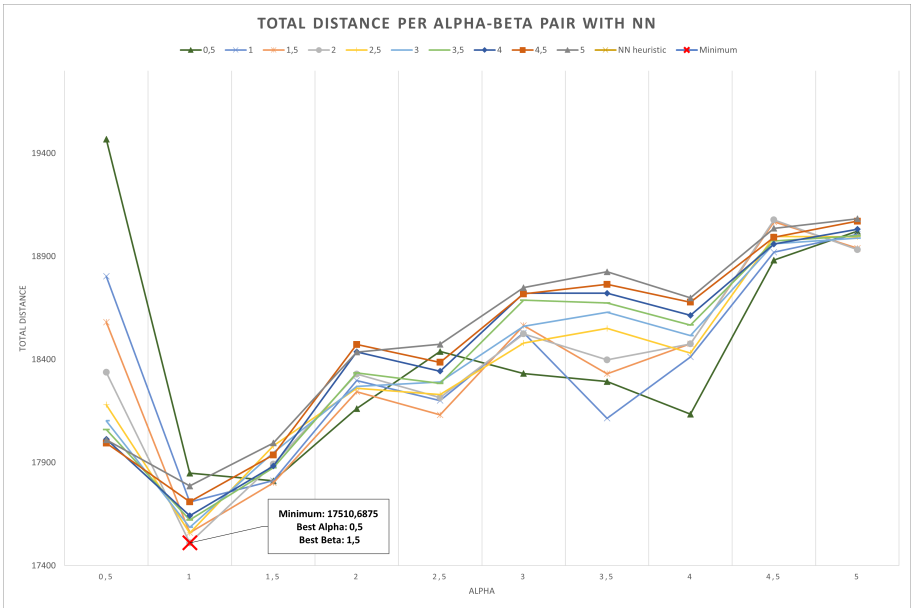

Figure 5.2: Line plot of average total distance calculated by the ACO with initial route using the different alpha-beta pairs for n = 25 for a combination of cars and tractors

5.2.2 ACO without Nearest Neighbour

For the ACO without the initial route, the alpha and beta values that reached the lowest average are: $\alpha = 1, \beta = 0.5$. A high alpha value and a relatively low beta value indicates that the ants will favour exploration using the pheromones predominantly as a guide instead of taking paths between nodes with a lower distance. The graph with all the different alpha-beta pairs and their average distances for the ACO with the initial route can be seen in Figure 5.3.

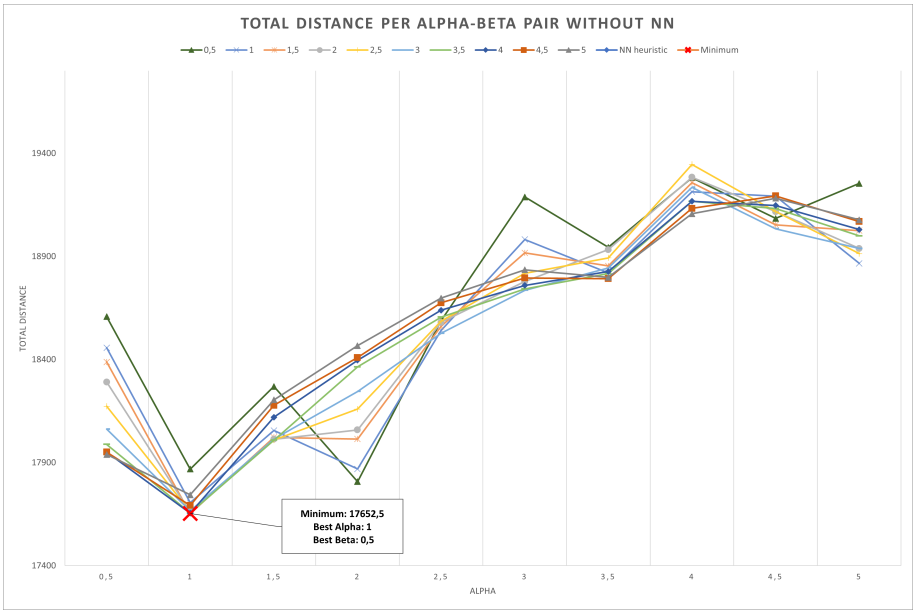

Figure 5.3: Line plot of average total distance calculated by the ACO without initial route using the different alpha-beta pairs for n = 25 for a combination of cars and tractors

5.3 Tested against

This section will elaborate on the two solution methods the designed solution model will be tested against. The first solution method the solution model will be tested against is the Moving ”South” heuristic, described in Chapter 2, which is made to represent the current route planning method of the city of the Hague. Secondly, the ACO will be tested against an ILP, which is there to represent to the most ideal scenario.

The ILP is used to calculate the optimal value. It is used as a benchmark of the quality of the solution found by the solution model. The ILP is modelled using the Miller-Tucker-Zemlin formulation described in Chapter 4 in Python. Furthermore, Gurobi is used to solve the model. For the Gurobi solver, the maximum computation time has been set to 120 seconds. If, after 120 seconds, the Gurobi solver still has not found a solution. The objective bound is taken for the total distance or duration calculated from the ILP.

5.4 Experiments

This section will cover the results of the different experiments. The abbreviations in the tables containing the results of the experiments refer to the following:

- MS = Moving South Heuristic
- NN = Nearest Neighbour Heuristic
- ACO = General Ant Colony Optimization Metaheuristic
- ACO + NN = Ant Colony Optimization Metaheuristic with initial Route generated by the Nearest Neighbour Heuristic
- ILP = Integer Linear Program
- Δ% = Difference in percentage

5.4.1 Tractors

The first experiment is to create a route for customers only using tractors to irrigate the sensors. As described in Section 5.1, the data used for the experiments is the created data of the city of the Hague. Furthermore, the experiments are run for several instances of sensor sizes. The optimization is done for driving duration and total distance. All different solution methods have been tested on the same dataset.

5.4.1.1 Total distance

The results of the experiments can be seen in Table 5.1. From the experiments on minimizing distance, it becomes apparent that all three proposed heuristic route planning approaches are better approaches than the current route planning strategy. The general ACO is able to produce solutions that are 20.12% (N = 10), 34.20% (N = 25), 28.58% (N = 50), and 25.42% (N = 100) better than the current strategy, while the ACO + NN produces solutions that are 20.44% (N = 10), 32.90% (N = 25), 27.44% (N = 50), and 24.00% (N = 100) better than the current route planning strategy.

Furthermore, it can be seen that the general ACO outperforms the ACO + NN by approximately 2% for all N except for N = 10. Furthermore, it can be seen that the solution of the ILP, which is assumed to be optimal, provides results that are just 1.66% (N = 10), 3.00% (N = 25) better than

the ACO solution. For N = 50, the ACO outperforms the ILP by 5.75% (N = 50), with no results found for N = 100. Here, it can be seen that when N increases, the gap with ACOs decreases to the point that the ILP cannot find a solution better than the ACO during its computational time. Furthermore, it can be seen that the improvements the ACO + NN has on the NN range from approximately 2 to 5% for all N except N = 25, where the difference jumps to almost 15%. The general ACO's improvements on the NN range from approximately 2 to 7% for all N except N = 25, where the improvement is almost 16%. There are, however, significant differences in computation times when comparing the ACOs and ILP with the NN and MS. The Moving "South" heuristic and the Nearest Neighbour heuristic both achieve sub-second computational times, whereas the ACOs and the ILP both have computational times of several seconds up to several hours.

	N	10	25	50	100
MS	Distance (meters)	125635	330528	576982	1106769
	time (s)	0,002	0,002	0,003	0,003
NN	Distance (meters)	102024	258153	440413	885097
	time (s)	0,01	0,04	0,13	0,73
	Δ% MS	-18,79%	-21,90%	-23,67%	-20,03%
ACO	Distance (meters)	100358	217474	410547	825459
	time (s)	1,04	21,71	233,36	3716,53
	Δ% MS	-20,12%	-34,20%	-28,85%	-25,42%
	Δ% NN	-1,63%	-15,76%	-6,78%	-6,74%
	Δ% NN + ACO	0,40%	-1,94%	-1,94%	-1,86%
	Δ% ILP	1,66%	3,00%	-5,75%	-
ACO+NN	Distance (meters)	99957	221778	418668	841117
	time (s)	1,05	22,22	234,01	4010,79
	Δ% MS	-20,44%	-32,90%	-27,44%	-24,00%
	Δ% NN	-2,03%	-14,09%	-4,94%	-4,97%
	Δ% ACO	-0,40%	1,98%	1,98%	1,90%
	Δ% ILP	1,26%	5,03%	-3,89%	-
ILP	Distance (meters)	98715	211147	435600	-
	Best bound	89889	88513	90236	-
	time (s)	6,08	120,33	121,99	133,35
	Δ% MS	-21,43%	-36,12%	-24,50%	-
	Δ% NN	-3,24%	-18,21%	-1,09%	-
	Δ% NN + ACO	-1,26%	-5,03%	3,89%	-
	Δ% ACO	-1,66%	-3,00%	5,75%	-

Table 5.1: Comparison of different optimization based on distance for Tractors.

5.4.1.2 Total driving duration

From the experiments on minimizing total driving duration, it becomes apparent that the ACO outperforms all the other heuristics. As can be seen in Table 5.2 the general ACO is able to produce solutions that are 20.01% (N = 10), 33.90% (N = 25), 28.19% (N = 50), and 25.10% (N = 100) better than the current strategy, while the ACO + NN produces solutions that are 19.78% (N = 10), 33.44% (N = 25), 26.95% (N = 50) and 23.54% (N = 100) better than the current solution strategy. Furthermore, the ACO + NN performs 0.28% (N = 10), 0.82% (N = 25), 1.71% (N = 50), and 2.05% (N = 100) worse than the general ACO.

It can be seen that the solution of the ILP, which is assumed to be optimal, provides the following improvement on the ACO: 1.14% (N = 10), 0.70% (N = 25). It furthermore provides an improve-

ment of 1.42% (N = 10) and 1.53% (N = 25). For N = 50, the general ACO and ACO + NN outperform the ILP by 12.04% and 10,51% (N = 50), respectively. There are no results found for N = 100 by the ILP. Furthermore, it can be seen that the NN is outperformed by the general ACO by approximately 2 to 7% for all N except N = 25, where it is outperformed by almost 16% and outperformed by the ACO + NN by approximately 2 to 5% for all N except N = 25, where it is outperformed by almost 15%. There are, however, significant differences in computation times when comparing the ACOs and ILP with the NN and MS. The Moving ”South” heuristic and the Nearest Neighbour heuristic both achieve sub-second computational times, whereas the ACOs and the ILP both have computational times of several seconds up to several hours.

	N	10	25	50	100
MS	Duration (minutes)	15501	41023	71692	137531
	time (s)	0,003	0,002	0,003	0,004
NN	Duration (minutes)	12627	32054	55262	110799
	time (s)	0,01	0,05	0,13	0,83
	Δ% MS	-18,54%	-21,86%	-22,92%	-19,44%
ACO	Duration (minutes)	12399,5	27078,5	51480	103005
	time (s)	1,03	22,07	215,19	3633,92
	Δ% MS	-20,01%	-33,99%	-28,19%	-25,10%
	Δ% NN	-1,80%	-15,52%	-6,84%	-7,03%
	Δ% NN + ACO	-0,28%	-0,82%	-1,71%	-2,05%
	Δ% ILP	1,14%	0,70%	-12,04%	-
ACO+NN	Duration (minutes)	12434,3	27303,5	52374,5	105162
	time (s)	1,04	22,28	214,03	3773,91
	Δ% MS	-19,78%	-33,44%	-26,95%	-23,54%
	Δ% NN	-1,53%	-14,82%	-5,23%	-5,09%
	Δ% ACO	0,28%	0,83%	1,74%	2,09%
	Δ% ILP	1,42%	1,53%	-10,51%	-
ILP	Duration (minutes)	12260	26891	58525	-
	Best bound	11204	10884	11548	-
	time (s)	6,45	120,30	121,80	133,33
	Δ% MS	-20,91%	-34,45%	-18,37%	-
	Δ% NN	-2,91%	-16,11%	5,90%	-
	Δ% NN + ACO	-1,42%	-1,53%	10,51%	-
	Δ% ACO	-1,14%	-0,70%	12,04%	-

Table 5.2: Comparison of different optimization based on driving duration for Tractors.

5.4.2 Cars

The second experiment is to create a route for customers only using cars to irrigate the sensors. The data used for the experiments is described in Section 5.1. Furthermore, the experiments are run for several instances of sensor sizes. The optimization is done for driving duration and total distance, and all different solution methods have been tested on the same dataset.

5.4.2.1 Total distance

The results of the experiments with regard to minimizing the total distance can be seen in Table 5.3. The table shows that all three proposed heuristic route planning approaches are better than the current route planning strategy. The general ACO is able to produce solutions that are 29.38% (N = 10), 29.58% (N = 25), 29.56% (N = 50), and 21.10% (N = 100) better than the current strategy, while the ACO + NN produces solutions that are 29.38% (N = 10), 29.34% (N

= 25), 27.74% (N = 50), and 17.48% (N = 100) better than the current route planning strategy. Furthermore, it can be seen that the general ACO outperforms the ACO + NN by 0% (N = 10), 0.34% (N = 25), 2.51% (N = 50), 4.39% (N = 100).

The table also shows the solutions of the ILP, which are assumed to be optimal. It shows no improvement on the ACO for N = 10. However, this is not the case for N = 25, 50 and 100, where the ILP improves 0.79% (N = 25) and 3.91% (N = 50) on the general ACO solution, with no results found for N = 100. In the case of the ACO + NN the ILP improves by 0% (N = 10), 1,13% (N = 25), and 6,58% (N = 50). Furthermore, it can be seen that the general ACO provides better results than the ACO + NN for all N except N = 10, where the difference between them is 0%. For N = 25, 50 and 100, the improvements are 0.34% (N = 25), 2.51% (N= 50) and 4.39% (N = 100). Moreover, it can be seen that the gap improvement of the general ACO on the NN heuristic is 6.46% (N = 10), 14.90% (N = 25), 8.21% (N = 50), 8.89% (N = 100). For the ACO + NN, the improvements on the NN are 6.47% (N = 10), 14.61% (N = 25), 5.85% (N = 50), 4.71% (N = 100).There are, however, significant differences in computation times when comparing the ACOs and ILP with the NN and MS. The Moving ”South” heuristic and the Nearest Neighbour heuristic both achieve sub-second computational times, whereas the ACOs and the ILP both have computational times of several seconds up to several hours.

	N	10	25	50	100
MS	Distance (meters)	134356	288883	563664	948895
	time (s)	0,003	0,003	0,003	0,003
NN	Distance (meters)	101441	239053	432567	821738
	time (s)	0,01	0,05	0,14	0,77
	Δ% MS	-24,50%	-17,25%	-23,26%	-13,40%
ACO	Distance (meters)	94884,5	203438	397069	748674
	time (s)	2,10	29,74	329,39	4632,18
	Δ% MS	-29,38%	-29,58%	-29,56%	-21,10%
	Δ% NN	-6,46%	-14,90%	-8,21%	-8,89%
	Δ% NN + ACO	0,00%	-0,34%	-2,51%	-4,39%
	Δ% ILP	0,00%	0,79%	3,91%	-
ACO+NN	Distance (meters)	94882	204136	407278	783065
	time (s)	2,06	29,56	335,65	4739,97
	Δ% MS	-29,38%	-29,34%	-27,74%	-17,48%
	Δ% NN	-6,47%	-14,61%	-5,85%	-4,71%
	Δ% ACO	0,00%	0,34%	2,51%	4,39%
	Δ% ILP	0,00%	1,13%	6,58%	-
ILP	Distance (meters)	94882	201849	382124	-
	Best bound	89490	93848	115367	-
	time (s)	3,04	120,47	122,26	134,94
	Δ% MS	-29,38%	-30,13%	-32,21%	-
	Δ% NN	-6,47%	-15,56%	-11,66%	-
	Δ% NN + ACO	0,00%	-1,13%	-6,58%	-
	Δ% ACO	0,00%	-0,79%	-3,91%	-

Table 5.3: Comparison of different optimization based on distance for Cars.

5.4.2.2 Total duration

The results of the experiments minimizing the total duration can be seen in Table 5.4. From the experiments on minimizing duration, it becomes apparent that all three proposed heuristic route planning approaches are better approaches than the current route planning strategy. The

general ACO is able to produce solutions that are 29.63% (N = 10), 28.17% (N = 25), 28.09% (N = 50), and 19.38% (N = 100) better than the current strategy, while the ACO + NN produces solutions that are 29.62% (N = 10), 37.47% (N = 25), 34.22% (N = 50), and 19.72% (N = 100) better than the current route planning strategy.

Furthermore, it can be seen that the general ACO outperforms the ACO + NN by 0.02% (N = 10), 1.25% (N = 25), 3.48% (N = 50), and 3.48% (N = 100). Furthermore, it can be seen that the solution of the ILP, which is assumed to be optimal, provides results that are no improvement upon the ACO for N = 50. It can be seen that the general ACO outperforms the ILP by 2.76% (N =50). However, for N = 10 and 25, it improves 0.02% (N = 10), 2.92% (N = 25), and for N = 100 no result is found within the time limit. For the ACO + NN, the ILP outperforms it by 0.04% (N = 10), 4.23% (N = 25) and 0.75% (N = 50). Just as the case is with the general ACO, there is no result found by the ILP for N = 100. Furthermore, it can be seen that the Nearest Neighbour heuristic provides results close to the general ACO, where the ACO shows improvements of 6.32% (N = 10), 14.95% (N = 25), 8.47% (N = 50) and 8.87% (N = 100). For the ACO + NN, the improvements upon the NN are 6.31% (N = 10), 13.87% (N = 25), 5.17% (N = 50) and 5.59% (N = 100). There are, however, significant differences in computation times when comparing the ACOs and ILP with the NN and MS. The Moving ”South” heuristic and the Nearest Neighbour heuristic both achieve sub-second computational times, whereas the ACOs and the ILP both have computational times of several seconds up to several hours.

	N	10	25	50	100
MS	Duration (minutes)	10519	22392	44627	75861
	time (s)	0,003	0,003	0,003	0,004
NN	Duration (minutes)	7902	18912	35060	67119
	time (s)	0,01	0,03	0,16	0,92
	Δ% MS	-24.88%	-15.54%	-21.44%	-11.52%
ACO	Duration (minutes)	7402.5	16085	32090.8	61162.3
	time (s)	2.04	28.65	356.76	4801.81
	Δ% MS	-29.63%	-28.17%	-28.09%	-19.38%
	Δ% NN	-6.32%	-14.95%	-8.47%	-8.87%
	Δ% NN + ACO	-0.02%	-1.25%	-3.48%	-3.48%
	Δ% ILP	0.02%	2.92%	-2.76%	-
ACO+NN	Duration (minutes)	7403.75	16288.8	33248.5	63365.5
	time (s)	2.09	27.66	365.40	4847.66
	Δ% MS	-29.62%	-27.26%	-25.50%	-16.47%
	Δ% NN	-6.31%	-13.87%	-5.17%	-5.59%
	Δ% ACO	0.02%	1.27%	3.61%	3.60%
	Δ% ILP	0.04%	4.23%	0.75%	-
ILP	Duration (minutes)	7401	15628	33000	-
	Best bound	6725	8255	9868	-
	time (s)	3.14	120.33	122.62	138.08
	Δ% MS	-29.64%	-30.21%	-26.05%	-
	Δ% NN	-6.34%	-17.36%	-5.88%	-
	Δ% NN + ACO	-0.04%	-4.23%	-0.75%	-
	Δ% ACO	-0.02%	-2.92%	2.76%	-

Table 5.4: Comparison of different optimization based on duration for Cars

5.4.3 Combination of vehicles

The third experiment is to create a route for customers using both cars and tractors to irrigate the sensors. The data used for the experiments is described in Section 5.1. Furthermore, the experiments are run for several instances of sensor sizes. The optimization is done for driving duration and total distance, and all different solution methods have been tested on the same dataset.

5.4.3.1 Total distance

The results of the experiments with the objective of minimizing total distance can be seen in Table 5.5. From the experiments on minimizing distance, it becomes apparent that all three proposed heuristic route planning approaches are better approaches than the current route planning strategy. The general ACO is able to produce solutions that are 39.55% (N = 10), 35.92% (N = 25), 40.47% (N = 50), and 39.75% (N = 100) better than the current strategy, while the ACO + NN produces solutions that are 42.02% (N = 10), 34.65% (N = 25), 39.40% (N = 50), and 37.48% (N = 100) better than the current route planning strategy. Furthermore, it can be seen that the general ACO outperforms the ACO + NN for all N except for N = 10. For N = 10, the ACO + NN outperforms the general ACO by 4.26%. However, for N = 25, 50 and 100, the ACO outperforms the ACO + NN by 1.94% (N = 25), 1.76% (N = 50), 3.62% (N = 100)

For the ILP, it can be seen that the solution, which is assumed to be optimal, provides results that improve the ACO for all N. The improvements of the ILP on the general ACO are 9.39% (N = 10), 14.05% (N = 25), and 6.47% (N = 50). Since no value is found by the ILP in the set time limit for N = 100, there is also no improvement upon the ACOs for this N. The improvements of the ILP on the ACO + NN are 4.92% (N = 10), 16.31% (N = 25), and 8.38% (N = 50). Furthermore, it can be seen that for N = 10, the NN outperforms the general ACO by 1.87% (N = 10). However, the general ACO improves upon the NN for N = 25, 50 and 100. The improvements are 12.53% (N = 25), 17.23% (N = 50) and 17.69% (N = 100). For the ACO + NN, the improvements upon the NN are 2.29% (N = 10), 10.80% (N = 25), 15.74% (N = 50), and 14.59% (N = 100). There are, however, significant differences in computation times when comparing the ACOs and ILP with the NN and MS. The Moving ”South” heuristic and the Nearest Neighbour heuristic both achieve sub-second computational times, whereas the ACOs and the ILP both have computational times of several seconds up to several hours.

	N	10	25	50	100
MS	Distance (meters)	105096	250038	556907	1076161
	time (s)	0,003	0,003	0,006	0,004
NN	Distance (meters)	62360	183169	400544	787726
	time (s)	0,02	0,10	0,38	2,62
	Δ% MS	-40,66%	-26,74%	-28,08%	-26,80%
ACO	Distance (meters)	63525,5	160222	331549	648409
	time (s)	0,29	15,48	190,36	3243,62
	Δ% MS	-39,55%	-35,92%	-40,47%	-39,75%
	Δ% NN	1,87%	-12,53%	-17,23%	-17,69%
	Δ% NN + ACO	4,26%	-1,94%	-1,76%	-3,62%
	Δ% ILP	9,39%	14,05%	6,47%	-
ACO+NN	Distance (meters)	60931	163396	337502	672772
	time (s)	0,32	15,21	189,64	3430,39
	Δ% MS	-42,02%	-34,65%	-39,40%	-37,48%
	Δ% NN	-2,29%	-10,80%	-15,74%	-14,59%
	Δ% ACO	-4,26%	1,94%	1,76%	3,62%
	Δ% ILP	4,92%	16,31%	8,38%	-
ILP	Distance (meters)	58073	140478	311406	-
	Best bound	52284	78858	108901	-
	time (s)	2,42	120,33	121,71	132,61
	Δ% MS	-44,74%	-43,82%	-44,08%	-
	Δ% NN	-6,87%	-23,31%	-22,25%	-
	Δ% NN + ACO	-4,92%	-16,31%	-8,38%	-
	Δ% ACO	-9,39%	-14,05%	-6,47%	-

Table 5.5: Comparison of different optimization based on distance for a combination of Tractors and Cars.

5.4.3.2 Used vehicles

The average number of cars and tractors used to construct the routes with the objective of minimizing total distance can be seen in Table 5.6. Here, it can be seen that for all the heuristics apart from the MS, the number of tractors used is approximately twice the number of cars used. This makes sense since the tractors are able to visit more sensors in a single route.

	N	10	25	50	100
MS	Cars used	1	5	14	37
	tractors used	1	4	10	19
NN	Cars used	0	2	4	12
	tractors used	1	4	9	16
ACO	Cars used	0	2	4	9,25
	tractors used	1	4	8	16,25
ACO+NN	Cars used	0	2	4	9,25
	tractors used	1	4	8	16

Table 5.6: Number of vehicles used during optimization on distance

5.4.3.3 Total duration

The results of the experiments minimizing the total duration can be seen in Table 5.7. From the experiments on minimizing duration, it becomes apparent that all three proposed heuristic route planning approaches are better approaches than the current route planning strategy. The general ACO is able to produce solutions that are 32.81% (N = 10), 33.97% (N = 25), 38.06% (N = 50), and 36.86% (N = 100) better than the current strategy, while the ACO + NN produces solutions that are 34.98% (N = 10), 32.14% (N = 25), 37.48% (N = 50), and 34.72% (N = 100) better than the current route planning strategy. Furthermore, it can be seen that the general ACO is outperformed by the ACO + NN for N = 10 by 3.33% (N = 10). For N =25,50 and 100, the general ACO outperforms the ACO + NN by 2.70% (N = 25), 0.94% (N = 50), and 3.27% (N = 100).

In the case of the ILP, it can be seen that the solution, which is assumed to be optimal, provides results that are an improvement upon the ACO for all N. With the ILP outperforming the ACO by 8.27% (N = 10), 6.17% (N = 25) and 2.25% (N = 50). The ILP outperforms the ACO + NN by 4.77% (N = 10), 9.11% (N = 25), and 3.22% (N = 50). For N = 100, the ILP has been unable to find a solution in the given time limit. It can also be seen that the Nearest Neighbour heuristic provides results close to the ACO and the ACO + NN where the NN outperforms the general ACO for N = 10 by 1.60% (N = 10). For N = 25,50 and 100, the general ACO outperforms the NN by 5.10% (N = 25), 6.31% (N = 50), and 12.32% (N = 100). In the case of the ACO + NN, it improves upon the NN by 1.68% (N = 10), 2.47% (N = 25), 5.42% (N = 50), and 9.35% (N = 100). There are, however, significant differences in computation times when comparing the ACOs and ILP with the NN and MS. The Moving ”South” heuristic and the Nearest Neighbour heuristic achieve sub-second computational times, whereas the ACOs and the ILP have computational times of several seconds up to several hours.

	N	10	25	50	100
MS	Duration (minutes)	11509	28061	64629	119408
	time (s)	0,003	0,003	0,003	0,003
NN	Duration (minutes)	7611	19524	42725	85990
	time (s)	0,02	0,11	0,53	2,54
	Δ% MS	-33,87%	-30,42%	-33,89%	-27,99%
ACO	Duration (minutes)	7732.5	18528.8	40028.8	75397.8
	time (s)	0.55	16.25	181.36	3238.02
	Δ% MS	-32,81%	-33,97%	-38,06%	-36,86%
	Δ% NN	1.60%	-5.10%	-6.31%	-12.32%
	Δ% NN + ACO	3.33%	-2.70%	-0.94%	-3.27%
	Δ% ILP	8.27%	6.17%	2.25%	-
ACO+NN	Duration (minutes)	7483	19042.3	40408.8	77950.3
	time (s)	0.67	15.64	182.94	3279.19
	Δ% MS	-34,98%	-32,14%	-37,48%	-34,72%
	Δ% NN	-1.68%	-2.47%	-5.42%	-9.35%
	Δ% ACO	-3.33%	2.70%	0.94%	3.27%
	Δ% ILP	4.77%	9.11%	3.22%	-
ILP	Duration (minutes)	7142	17452	39148	-
	Best bound	6567	9270	13237	-
	time (s)	3.10	120.26	122.28	132.39
	Δ% MS	-37,94%	-37,81%	-39,43%	-
	Δ% NN	-6.16%	-10,61%	-8.37%	-
	Δ% NN + ACO	-4.77%	-9,11%	-3.22%	-
	Δ% ACO	-8.27%	-6.17%	-2.25%	-

Table 5.7: Comparison of different optimization based on duration for a combination of Tractors and Cars.

5.4.3.4 Used vehicles

The average number of cars and tractors used to construct the routes with the objective of minimizing the total driving duration can be seen in Table 5.8. Just as with the vehicles used for the route construction based on distance, it can be seen that for all the heuristics apart from the MS, the number of tractors used is approximately twice the number of cars used. This makes sense since the tractors are able to visit more sensors in a single route and, therefore, would be chosen more frequently than the car.

	N	10	25	50	100
MS	Cars used	1	5	14	37
	tractors used	1	4	10	19
NN	Cars used	1	2	5	12
	tractors used	1	4	8	16
ACO	Cars used	0	2	4,25	10,5
	tractors used	1	4	8	16
ACO+NN	Cars used	0	2	4	10,5
	tractors used	1	4	8	16,25

Table 5.8: Number of vehicles used during optimization on duration

5.4.4 Combination of tractors and cars, where tractors visit the sensors outside the city centre and cars the sensors in the city centre

The last experiment is creating a route for customers using cars and tractors to irrigate the sensors. However, in this experiment, two scenarios are tested against each other. The first is the scenario where the tractors only visit the sensors outside the city centre, and the cars only visit the sensors in the city centre. The second scenario is the case where both the sensors in the city centre and outside the city centre can be visited by both cars and tractors. The data used for the experiments is described in Section 5.1. Furthermore, the experiments are run for the sensor sizes N = 20, 50, and 100, and the optimization is done for driving duration and total distance. All different solution methods have been tested on the same dataset.

5.4.4.1 Total distance

The results of the experiment conducted on minimizing the total distance can be seen in Table 5.9. Here, it can be seen that for all N, the combined case, where both the tractors and the cars can visit all the sensors, is better than the case where the sensors are visited vehicle-dependent. The total distances found by the general ACO on the combined case are 29.23% (N = 20), 23.27% (N = 50), and 19.98% (N = 100) lower than the distances found by the general ACO on the separate case. The total distances found by the ACO + NN on the combined case are 29.58% (N = 20), 23.83% (N = 50) and 22.21% (N = 100) lower than the separate case. This means that, in general, the combined case results in routes that are approximately 20-30% better than the vehicle-dependent cases.

	N	20	50	100
NN	Distance (meters) Tractors + Cars	205281	652553	1483175
	Distance (meters) Combined Case	170957	539001	1211670
	time (s) Tractors + Cars	0,031266022	0,140705776	0,38916316
	time (s) Combined Case	0,081970692	0,612422705	3,543166113
	Δ% Combined Case	-16,72%	-17,40%	-18,31%
ACO	Distance (meters) Tractors + Cars	199805,6	579147,8	1343357,8
	Distance (meters) Combined Case	141395,4	444362,4	1074924
	time (s) Tractors + Cars	4,413877	96,08278	1176,705
	time (s) Combined Case	9,814246	370,6633	7357,221
	Δ% Combined Case	-29,23%	-23,27%	-19,98%
ACO + NN	Distance (meters) Tractors + Cars	201276,2	579378,4	1355044,8
	Distance (meters) Combined Case	141730,2	441300,2	1054074
	time (s) Tractors + Cars	4,192223	100,6075	1217,662
	time (s) Combined Case	9,946356	375,508	6866,73
	Δ% Combined Case	-29,58%	-23,83%	-22,21%

Table 5.9: Comparison of different optimization based on distance for the comparison between the case that tractors visit sensors outside of the centre and cars in the city centre and the combined case.

5.4.4.2 Vehicles used

As can be seen in Table 5.10 for all different optimization methods and all N, the combined case uses fewer vehicles than the separate case. More specifically, the combined case leads to a reduced number of cars being used while the usage of the tractors increases. This is in direct relation to the decrease in total distance. In the separate case, the increased usage of cars means more frequent visits to the depot, thereby resulting in extra distance. Since, in the combined case, the usage of cars decreases and the the usage of tractors increases, extraneous distance associated with visits to the depot decreases as the tractors can visit more sensors in a single route resulting in fewer depot visits and a lower total distance.

	N	20	50	100
NN	Tractors used (Cars + Tractors)	3	9	21
	Cars used (Cars + Tractors)	4	12	27
	Tractors used (Combined case)	4	10	22
	Cars used (Combined Case)	1	8	16
	Δ tractors Combined Case	1	1	1
	Δ cars Combined Case	-3	-4	-11
ACO	Tractors used (Cars + Tractors)	3	9,8	20
	Cars used (Cars + Tractors)	4	11	26
	Tractors used (Combined case)	4	11,2	23,2
	Cars used (Combined Case)	1	4	13
	Δ tractors Combined Case	1	1,4	3,2
	Δ cars Combined Case	-3	-7	-13
ACO + NN	Tractors used (Cars + Tractors)	3	9,8	20
	Cars used (Cars + Tractors)	4	11	26
	Tractors used (Combined case)	4	11,4	24,2
	Cars used (Combined Case)	1	3,8	11
	Δ tractors Combined Case	1	1,6	4,2
	Δ cars Combined Case	-3	-7,2	-15

Table 5.10: Comparison of the vehicles used during optimization based on the tractors visiting sensors outside of the city centre and cars in the city centre versus the combined case (distance).

5.4.4.3 Total driving duration

The results of the experiment conducted on minimizing the total duration can be seen in Table 5.11. Here, it can be seen that for all N, the combined case, where both the tractors and the cars can visit all the sensors, is better than the case where the sensors are visited vehicle-dependent. The total durations found by the general ACO on the combined case are 19.22% (N = 20), 10.89% (N = 50), and 10.68% (N = 100) lower than the distances found by the general ACO on the separate case. The total durations found by the ACO + NN on the combined case are 17.80% (N = 20), 12.13% (N = 50) and 12.44% (N = 100) lower than the separate case. This means that, in general, the combined case results in routes that are approximately 10-20% better than the vehicle-dependent cases.

	N	20	50	100
NN	Driving duration (minutes) Tractors + Cars	19597	59457	143599
	Driving duration (minutes) Combined Case	19924	56035	128739
	time (s) cars + tractors	0,029916143	0,097739792	0,398420429
	time (s) combined case	0,073803091	0,598010826	3,360398769
	∆% Combined Case	1,64%	-6,11%	-11,54%
ACO	Driving duration (minutes) Tractors + Cars	19393,8	55526,8	128028,4
	Driving duration (minutes) Combined Case	16267,4	50073,6	115671
	time (s) cars + tractors	3,509159	76,4	1125,367
	time (s) combined case	10,04111	366,8762	6612,207
	∆% Combined Case	-19,22%	-10,89%	-10,68%
ACO + NN	Driving duration (minutes) Tractors + Cars	19261,2	55856	129552,8
	Driving duration (minutes) Combined Case	16351,2	49812,6	115221,8
	time (s) cars + tractors	4,192223	100,6075	1217,662
	time (s) combined case	9,806349	371,344	6460,155
	∆% Combined Case	-17,80%	-12,13%	-12,44%

Table 5.11: Comparison of different optimization based on duration for the comparison between the case that tractors visit sensors outside of the centre and cars in the city centre and the combined case.

5.4.4.4 Vehicles used

As can be seen in Table 5.12 for all different optimization methods and all N, the combined case uses fewer vehicles than the separate case. More specifically, the combined case leads to fewer cars being used while the usage of tractors increases. Just as with the optimization based on minimizing distance, this directly relates to the decrease in total distance. In a separate case, the increased usage of cars means more frequent visits to the depot, thereby resulting in extra distance. Since, in the combined case, the usage of cars decreases and the usage of tractors increases, extraneous distance associated with visits to the depot decreases as the tractors can visit more sensors in a single route, resulting in fewer depot visits and a lower total distance.

	N	20	50	100
NN	Tractors used (Cars + Tractors)	3	9	22
	Cars used (Cars + Tractors)	4	11	26
	Tractors used (Combined case)	4	10	22
	Cars used (Combined Case)	1	8	16
	Δ tractors Combined Case	1	1	0
	Δ cars Combined Case	-3	-3	-10
ACO	Tractors used (Cars + Tractors)	3	10	19,8
	Cars used (Cars + Tractors)	4	11	26
	Tractors used (Combined case)	4	10,6	22,4
	Cars used (Combined Case)	1	5,8	14,2
	Δ tractors Combined Case	1	0,6	2,6
	Δ cars Combined Case	-3	-5,2	-11,8
ACO + NN	Tractors used (Cars + Tractors)	3	9,6	20
	Cars used (Cars + Tractors)	4	11	26
	Tractors used (Combined case)	4	10,8	23,4
	Cars used (Combined Case)	1	4,4	12,8
	Δ tractors Combined Case	1	1,2	3,4
	Δ cars Combined Case	-3	-6,6	-13,2

Table 5.12: Comparison of the vehicles used during optimization based on the tractors visiting sensors outside of the city centre and cars in the city centre versus the combined case (duration).

5.4.5 Solution model output

From the conducted experiments, several outputs have been generated. These outputs are the maps that the customers of Connected Green will be able to view to see the calculated routes. An example of such a map can be seen in Figure 5.4. It is a map of the city of the Hague, with combined vehicles and N = 100 optimised on driving duration. The map contains 26 routes to visit all the nodes, which are depicted in different colours on the map. Furthermore, each sensor is marked with its colour and position in the route order. For example, the sensor in position 7 of a route with a red line, will have a red marker with the number seven. All the maps are interactive so that the customers are able to click on the sensors to view the name of the sensor, zoom in and out, and deselect specific routes to see them more clearly.

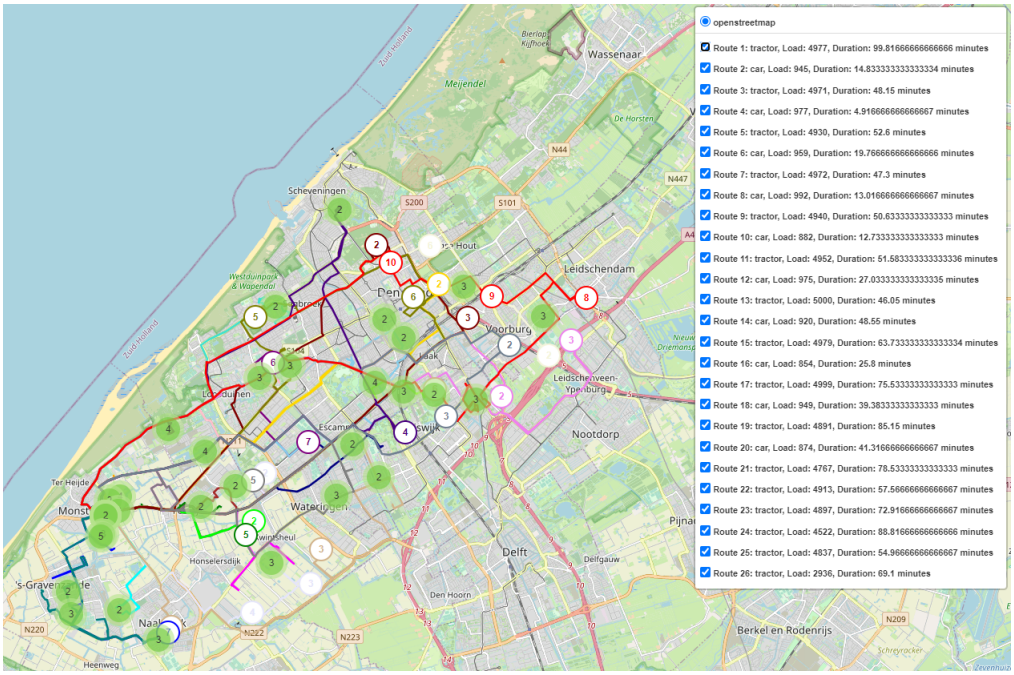

Figure 5.4: Output map of the solution model for the city of the Hague for combined vehicles optimised on driving duration for N = 100.

5.4.6 Experiment conclusion

From the experiments, three main cases can be distinguished: comparison between assigning vehicles to specific sensors, optimization based on distance, and optimization based on duration. This section will outline the main results from the experiments on these cases.

5.4.6.1 comparison between assigning vehicles to specific sensors

Lastly, the experiment which compared the situation where sensors in the city centre can only be visited by cars and sensors outside the city centre can only be visited by tractors. From the experiments, it is evident that for every optimization method, the combined case, where every vehicle can visit every sensor, provides better results. In the case of the general ACO, the improvements are 29.23% (N = 20), 23.27% (N = 50) and 19.98% (N = 100), with the average over all N being 24.16%. For the ACO + NN, the improvements are 29.58% (N = 20), 23.38% (N = 50) and 22.21% (N = 100), with the average over all N being 25.21%.

From this can be concluded that it is better to let all vehicle types visit all sensors and not assign specific vehicles to specific sensors.

5.4.6.2 Optimization on distance

First, the experiments were based on distance. From the experiments conducted, it can be concluded that both the ACO and the ACO + NN outperform both the current solution strategy and the Nearest Neighbour heuristic. Table 5.13 shows the average results from the experiments based on minimizing the total distance. Here, it can be seen that, on average, the general ACO outperforms the current solution strategy by 29,68% (N = 10), 33.23% (N = 25), 32.96% (N = 50) and 28.12% (N = 100). Averaging a 31.00% improvement over all N. In comparison, on average, the ACO + NN outperforms the current solution strategy by 30.61% (N = 10), 32.30% (N = 25), 31.53% (N = 50) and 25.60% (N = 100). Averaging a 30.01% improvement over all N.

In terms of computation times, the averages are almost identical, as can be seen in Table 5.13. Only for N = 100 are the differences greater than one second. Here, the ACO has a lower computational time than the ACO + NN, with the difference being at most 43.51 seconds. This is, in terms of percentages, a 1.07% difference.

In comparison with the ILP, the general ACO has, on average, a difference of 3.69% (N = 10), 5.95% (N = 25) and 1.54% (N = 50), averaging a 3.72% difference over all N. Whereas the ACO + NN, on average, has a difference of 2.06% (N = 10), 7.49% (N = 25) and 3.69% (N = 50), averaging a 4.42% difference over all N. Meaning that the general ACO is on average just 3.69% away from the optimal solution.

Considering everything, it can be concluded from the experiments on duration that, on average, the general ACO outperforms the ACO + NN.

	N	10	25	50	100
ACO	avg. Δ% MS	-29,68%	-33,23%	-32,96%	-28,12%
	avg. Δ% ACO + NN	1,55%	-1,41%	-2,07%	-3,38%
	avg. Δ% ILP	3,69%	5,95%	1,54%	-
	avg. time (s)	1,14	22,31	251,03	4019,91
ACO + NN	avg. Δ% MS	-30,61%	-32,30%	-31,53%	-25,60%
	avg. Δ% ILP	2,06%	7,49%	3,69%	-
	avg. time (s)	1,15	22,33	253,10	4063,42

Table 5.13: Average experiment results for optimization based on distance.

5.4.6.3 Optimization on duration

Secondly, the experiments based on duration. From the conducted experiments, it can be concluded that both the ACO and the ACO + NN outperform both the current solution strategy and the Nearest Neighbour heuristic. In Table 5.14, the average results from the experiments based on minimizing the total distance can be seen. From the table, it can be seen that, on average, the general ACO outperforms the current solution strategy by 27,48% (N = 10), 32.04% (N = 25), 31.45% (N = 50) and 28.76% (N = 100). Resulting in an average of 29.93% improvement over all N. In comparison, on average, the ACO + NN outperforms the current solution strategy by 28.13% (N = 10), 30.95% (N = 25), 29.97% (N = 50) and 26.32% (N = 100). Averaging a 28.84% improvement over all N.

In terms of computation times, the averages are similar for the ACO and the ACO + NN. As can be seen in Table 5.14, the only real differences are for N = 50 and 100. For N = 50, the ACO + NN is approximately 3 seconds slower, which, in terms of percentages, is a 1.19% difference. Whereas with N = 100, the ACO + NN is 196.28 seconds slower, which is a 4.83% difference

in percentages.

In comparison with the ILP, the general ACO produces solutions that are, on average, 3.14% (N = 10), and 3.26% (N = 25) worse than the ILP. For N = 50, the average improvement of the ACO on the ILP is 4.18% (N = 50). Averaging 0.74% difference with the ILP over all N. Whereas the ACO + NN, on average, produces solutions that are 2.08% (N = 10), 4.96% (N = 25) worse than the ILP. For N = 50, the ACO + NN outperforms the ILP by 2.18% (N = 50). Averaging 1.62% difference with the ILP over all N.

Considering everything, it can be concluded from the experiments on duration that, on average, the general ACO outperforms the ACO + NN.

	N	10	25	50	100
ACO	avg. Δ% MS	-27,48%	-32,04%	-31,45%	-28,76%
	avg. Δ% ACO + NN	1,01%	-1,59%	-2,04%	-3,29%
	avg. Δ% ILP	3,14%	3,26%	-4,18%	-
	avg. time (s)	1,21	22,32	251,10	3864,11
ACO + NN	avg. Δ% MS	-28,13%	-30,95%	-29,97%	-26,32%
	avg. Δ% ILP	2,08%	4,96%	-2,18%	-
	avg. time (s)	1,27	21,86	254,12	4060,39

Table 5.14: Average experiment results for optimization based on duration.

Considering everything, it can be concluded from the experiments that, on average, the general ACO outperforms the ACO + NN. Furthermore, it can be concluded that the general ACO is the best fit for the solution model even though with a larger N, the computational times increase significantly. However, in the case of Connected Green, the current customers have a maximum of 20 sensors, which means that the ACO can still be used in the solution model.

5.5 Evaluation plan and implementation plan

This section will elaborate on the evaluation and implementation plan, that Connected Green will be able to follow to evaluate and implement the model accordingly. First, an evaluation plan is proposed on how the solution model can be evaluated by the customers. Afterwards, an implementation plan is proposed on how the model can be correctly implemented.

5.5.1 Implementation plan

To be able to effectively implement the solution model in the current dashboard of Connected Green, several elements will have to be edited. The things that need to be revised are:

1. Hosting Graphhopper Server:
   Currently, the solution model uses a locally hosted Graphhopper API to calculate all the different distance matrices and route data used for mapping. For the customers to be able to use the model this API will need to be hosted on a server by Connected Green.

2. Precalculation of distance matrices:
   As the locations of the sensors do not change frequently, it is possible to calculate and store the distance matrices with the locations from and to all sensors beforehand. This will circumvent the solution model needing to calculate the distance matrices for the to-be-visited points by creating a custom distance matrix copying the already calculated distances or durations from and to each point. This will speed up the solution model as the calculation of distant matrices takes a significant time.

3. Predetermination of route data:
   As the route between two nodes is always the same, the route data for this route can be predetermined and saved on the servers of Connected Green. This will speed up the solution model as it does not need to determine the route data each time an irrigation route is calculated.

4. Making a custom API:
   Currently, two APIs need to be called to load in all relevant customer data from the dashboard. In the case that Connected Green creates a new API which loads in all relevant data at once, the solution model can be sped up.

5. Making a demand input section in the dashboard:
   As the demand of the customers is unknown, this will need to be inputted by the customers. To be able to input this Connected Green should make an input section, where the customers can provide them with the demand per sensor. This data should be saved to the server to be used to create an average demand for new customers or customers who do not want to input their own demand.

6. Precalculating irrigation routes:
   As underlined by the experiments in Section 5.4, when the number of sensors that need irrigation grows the ACO will take a significant time to determine the optimal route. Therefore, Connected Green should predetermine routes that include all sensors and save them. In this way, the solution model would not need to calculate these routes again.

5.5.2 Evaluation plan

This research focuses on creating a solution model that can be used by the customers of Connected Green to calculate irrigation routes. Currently, there is little to no data available on how the customers are currently doing this. Therefore to evaluate the solution model, it is first effective to evaluate and test it using a real customer scenario. The most effective way this can be done, is to use a customer with a high number of sensors to test the solution model in a beta test[1]. One of the customers who would be a good fit for the beta test is the municipality of the Hague. This municipality has one of the highest number of sensors among the customers of Connected Green and has indicated to use the sensors for route planning. The evaluation plan will consist of the following steps:

1. Implement the model into the dashboard:
   First, it is necessary to implement the solution model into the dashboard, while making it not available for the customers yet.

2. Make the solution model available for the customers that will be beta testing:
   Secondly, after the customers have been selected for the beta testing, it is necessary to make sure they are the only customers able to use and see the model.

3. Let the customers use the model for a set period:
   Within this step, it is very important that the customer log the problems and inconsistencies noticed while using the solution model. Furthermore, it is important that the customers keep track of the driven routes and the water demand of the sensors during the beta test.

4. Meet with the customers to discuss all flaws and inconsistencies:
   Collect all the data the customers have created, along with their feedback on the use of the model.

[1] a field test of the beta version of a product (such as software) especially by testers outside the company developing it that is conducted prior to commercial release

5. Improve the model using the customer feedback: Use the collected data and feedback to improve the model and remove inconsistencies and flaws.

5.6 Conclusion

This chapter describes how the solution model and all proposed optimization methods are tested against a simulated scenario and how Connected Green can implement and evaluate the solution model with their own customers. First, the design of the experiments is elaborated on. Starting with a description of the data used within the experiments, the creation of the water demand dataset and on which probability distribution it has been modelled. It then explains how the sensor location data has been simulated and the dataset has been created. Furthermore, the different experiment scenarios, Key Performance Indicators and validation of the experiments are discussed. The setting of parameters for the proposed Ant Colony Optimization algorithms is also clarified. The chapter then discusses how the proposed heuristics are tested against the Moving ”South” heuristic and the ILP, followed by a comprehensive discussion of the experiment results and the solution model output. Lastly, the implementation and evaluation plan Connected Green can use to successfully implement and evaluate the solution model within the dashboard is outlined. The evaluation plan elaborates on the beta testing Connected Green should perform with an existing customer to identify any inconsistencies or flaws within the solution model. The implementation plan covers the necessary steps to successfully implement the solution model and make it work as efficiently as possible with the dashboard of Connected Green.

6 CONCLUSION

This final chapter concludes the research that has been performed at Connected Green, and has been reported on in the previous chapters. Section 6.1, will answer and motivate the main research question formulated in Chapter 1. In Section 6.2, the recommendation for Connected Green will be discussed. Section 6.3 discusses the further research that can be conducted at Connected Green. In Section 6.4, the theoretical and practical contributions are elaborated on. Lastly, Section 6.5 will both discuss the limitations and reflect on the solution model.

6.1 Conclusions

This section will answer and motivate the research question stated in Chapter 1. The main research question is stated as:

What is the most effective approach to develop an optimal route planning model, and how can it be leveraged by Connected Green to minimize the costs incurred by their customers?

The answer to the research question has been derived by answering all the sub-research questions from the previous chapters. First of all, in Chapter 2, a contextual analysis has been performed to gather information about the current route planning strategies, the customers, the requirements of the customers, all constraining factors, important features of a solution model, and the current dashboard of Connected Green. Furthermore, a current route planning strategy has been converted in to algorithm to be used for the testing of the solution model.

In Chapter 3, a literature study has been conducted with the aim of finding the best solution method for route planning in the case of Connected Green. To accomplish this, the Vehicle Routing Problem that Connected Green's customers are experiencing has been identified to be a Capacitated Vehicle Routing Problem. From the literature, it was concluded that a combination of an initial route creation by the Nearest Neighbourhood heuristic would be the best fit for the solution model.

Chapter 4 describes the proposed solution model for Connected Green. The chapter discusses the Vehicle Routing Problem that the customers are experiencing and describes it as an Integer Linear Program according to the Miller-Tucker-Zemlin formulation (Miller et al., 1960). Describing the parameters, objective function, and constraints. Furthermore, the requirements and assumptions of the solution model are elaborated on. The chapter then discusses the proposed heuristic and metaheuristic in more detail, explaining how the algorithms of the solution model work. After the inner workings of the algorithms have been explained, the chapter then discusses the functionality of both the function that is used to create the distance matrices which take the road accessibilities into account and the function to visualize the calculated routes.

In Chapter 5, the proposed algorithms were tested against various solution methods, including the current solution strategy. With the experiments focusing on minimizing total distance, the ACO + NN algorithm surpassed the current strategy by 30.01% but was 4.42% removed from the optimal. The general ACO showed a 31.00% improvement with a 3.69% gap from the best solution. In terms of speed, the ACO was 1.07% faster than ACO + NN. For driving duration minimization, the ACO + NN and general ACO improved upon the current strategy by 28.84% and 29.93%, respectively, with the ACO being 1.19% faster. Furthermore, the ACO + NN had an average gap with the optimal solution of 4.42% and 1.62%.

Another experiment compared scenarios where sensors are matched with specific vehicles versus the case where any vehicle could visit any sensor. From the results it became evident that the unrestricted approach was superior, improving performance by 24.16% by using the general ACO and 25.21% by using the ACO + NN.

Overall, the general ACO is superior in optimization and computational efficiency compared to ACO + NN. Despite the ACO's longer computation times with larger sensor counts, the ACO is a good fit for the solution model as the longer computational times should not be a concern for Connected Green, since the current customers have a maximum of 20 sensors per project, of which not all will need irrigation simultaneously. Furthermore, this chapter also provides an evaluation and implementation plan for the solution model, which Connected Green will be able to use to implement and evaluate the solution model with customers successfully.

Taking all chapters into account, the answer to the research question is to use an Ant Colony Optimization to create near-optimal routes for the customers so that the total distance or driving duration for the customers is significantly decreased and, therefore, the associated costs of these irrigation routes decrease accordingly.

6.2 Recommendations

After conducting the research and creating the solution model, several recommendations for Connected Green have been drafted.

The first recommendation is to make it possible for the customers to select the starting location for the routes. Currently, the solution model uses the project address of the customers as the starting location, which most customers indicated as being correct. However, some customers indicated that the starting locations will differ per week, and it could be either at a location of their own customers or along a route. Therefore, to have a solution model which can be used by all customers, it may prove convenient to let the customers choose their own starting locations.

Secondly, as the demand dataset has been created using a probability distribution, which in all likelihood will not correctly represent the reality of the customers, it is recommended to let the customers input their own data which Connected Green should also store. This data can then be used to create correct routes for the customers and utilised for the correct parameter settings for the Ant Colony Optimization.

Thirdly, it is recommended for Connected Green to create an API which can be used to retrieve all relevant information out of the dashboard that will be needed to calculate the routes. This would decrease computational times as currently, there are two different APIs needed to retrieve all relevant information.

Fourthly, to further decrease the computational times of the solution model it is recommended to store and calculate the distance matrices containing all the sensors of a project. This way, the distance matrix calculations have already been made beforehand and will not be needed to be done again.

Lastly, it is recommended to beta-test the solution model with several customers. In this way, inconsistencies and flaws can be identified which can then be corrected.

6.3 Further research

In this research, a solution model is developed in Python to automatically calculate optimal routes for the customers of Connected Green. However, this solution model can still be improved. In this section, some suggestions regarding further research are discussed.

Firstly, the sensors of Connected Green monitor the moisture levels in the ground. In further research, it may prove valuable to construct a model which can predict the water demand at the sensors. Integrating this with the solution model transforms it into a solution model which can be used to create preventive irrigation routes in which the plants can be watered at the optimal time.

Secondly, further research into the algorithms used within the solution model could prove fruitful. Currently, it uses a dynamic vehicle selection procedure for heterogeneous vehicle fleets to select the next vehicle to be used for the irrigation routes. However, it is not certain that this procedure reflects reality in the best way. In beta testing with customers, this selection procedure could be checked against real scenarios and improved if necessary.

Thirdly, the Ant Colony Optimization Algorithm, which improves upon an initial route generated by the Nearest Neighbour heuristic, uses the ”initial pheromone” parameter. In this research, this parameter was set to a value of 1.5. However, adjusting this value might yield improved results from the algorithm. Thus, further exploration of the optimal setting for the ”initial pheromone” parameter is recommended.

Lastly, since the customers of Connected Green are far from homogeneous it is recommended to also research cases wherein the customers have several depot locations. This can further improve the generalizability of the model for all possible customers of Connected Green.

6.4 Contributions

This section elaborates on the theoretical and practical contributions of this research.

6.4.1 Theoretical contribution

Within this research, the contribution to the literature is on the topic of Vehicle Routing Problems, more specifically the Capacitated Vehicle Routing Problem and its solution methods. Within the research, the Nearest Neighbour heuristic and Ant Colony Optimization metaheuristic were selected and combined to be used within the solution model. The combined case, where the Ant Colony Optimization takes the route found by the Nearest Neighbour heuristic as an initial route has been tested against a general Ant Colony Optimization on the case of the Heterogeneous Capacitated Vehicle Routing problem. This has been shown to be inferior to the general Ant Colony Optimization Algorithm.

6.4.2 Practical contribution

This research has been performed at Connected Green. The practical contribution for the company is the solution model that has been described in Chapter 4, and the evaluation and implementation plans described in Chapter 5. The solution model can be implemented into the dashboard of Connected Green using the implementation plan. By testing the solution model on a small number of customers using beta testing, the model can be evaluated and edited if needed. After which it can be available for all customers to calculate better irrigation routes.

6.5 Limitations

This section will cover the limitations of the created solution model, Connected Green should take these limitations into account when the solution model will be implemented into the dashboard.

First, since there was no dataset available for the experiments, to still test the solution model a dataset has been created. This dataset has been created under several assumptions. For instance, the probability distribution used to create the demand for the sensors is a triangular distribution. This distribution may not represent a real-life case well enough, however, the demand created by the distribution was used to set the parameters of the Ant Colony Optimization. Since the real demand of the customers will more than likely not follow this distribution, the parameters of the current Ant Colony Optimization may not be the optimal and therefore could create solutions of less value than would be possible.

Secondly, because of the lack of data, the service times at the sensors are not taken into consideration. This could lead to inconsistencies with optimal routes since it could be more beneficial to take another route when service times are taken into consideration.

Finally, the computation time of the Ant Colony Optimization algorithm was shown to significantly increase when the number of sensors increases. The solution model could be a bad fit for customers with more than 50 sensors. However, in this case, the Nearest Neighbour algorithm can be used to create routes that are close to the routes the Ant Colony Optimization will generate in a fraction of the time it takes.

References

Avdoshin, S., & Beresneva, E. (2019). Constructive heuristics for capacitated vehicle routing problem: a comparative study. *Proceedings of the Institute for System Programming of the RAS*, *31*, 145-156. doi: 10.15514/ISPRAS-2019-31(3)-12

Ayanso, A. (2014). *Business and technology trends in social crm*. IGI Global. doi: 10.4018/978-1-4666-6547-7.CH013

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Source: Operations Research*, *12*, 568-581.

Collins, T. D. (1998). The application of software visualization technology to evolutionary computation: A case study in genetic algorithms artificial intelligence. Retrieved from http://dx.doi.org/doi:10.21954/ou.ro.00006fa3http://people.kmi.open.ac.uk/trevor/archive/thesis/ doi: 10.21954/ou.ro.00006fa3

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Source: The Journal of the Operational Research Society*, *53*, 512-522. Retrieved from www.palgrave-journals.com/jors doi: 10.1057/palgrave/jors/2601319

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. P., & Vigo, D. (2007). Vehicle routing. , *14*. doi: 10.1016/S0927-0507(06)14006-2

Corona-Gutiérrez, K., Nucamendi-Guillén, S., & Lalla-Ruiz, E. (2022). Vehicle routing with cumulative objectives: A state of the art and analysis. Retrieved from http://creativecommons.org/licenses/by/4.0/ doi: 10.1016/j.cie.2022.108054

Dantzig, & Ramser. (1959). The truck dispatching problem. *Management Science*, 80- 91. Retrieved from https://ut.on.worldcat.org/search?queryString=vehicle20routing\&clusterResults=true\&groupVariantRecords=false

Dictionary, C. (n.d.). *Open-source | english meaning - cambridge dictionary.* Retrieved from https://dictionary.cambridge.org/dictionary/english/open-source

Dorigo, M., & Caro, G. D. (1999). Ant colony optimization: A new meta-heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, *2*, 1470-1477. doi: 10.1109/CEC.1999.782657

Een europese green deal. (n.d.). Retrieved from \url{https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal_nl},year={2023},

Fred, G. (1990). Tabu search: A tutorial. *The Practice of Mathematical Programming*, *20*, 74-94. Retrieved from https://www.jstor.org/stable/25061372

Heerkens, H., van. Winden, A., & Tjooitink, J.-W. (2017). *Solving managerial problems systematically*. Noordhoff Uitgevers :.

Hoffman, K. L., Padberg, M., & Rinaldi, G. (2001). Traveling salesman problem. doi: 10.1007/1-4020-0611-X_1068

Johnson, D. S., & Mcgeoch, L. A. (2007). Experimental analysis of heuristics for the stsp. , 369-443. Retrieved from http://dimacs.rutgers.edu/Challenges/.

Kissell, R., & Poserina, J. (2017, 1). Advanced math and statistics. *Optimal Sports Math, Statistics, and Fantasy*, 103-135. doi: 10.1016/B978-0-12-805163-4.00004-9

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J. M., & Brunese, P. A. (2019, 3). Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, *129*, 14-30. doi: 10.1016/J.CIE.2019.01.020

Lahyani, R., Khemakhem, M., & Semet, F. (2015, 2). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, *241*, 1-14. doi: 10.1016/J.EJOR.2014.07.048

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43*, 408-416. doi: 10.1287/TRSC.1090.0301

Lenstra, J. K., & Kan, A. H. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, *11*, 221-227. doi: 10.1002/NET.3230110211

Li, J., Ma, Y., Gao, R., Cao, Z., Lim, A., Song, W., & Zhang, J. (2021). Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. *IEEE TRANSACTIONS ON CYBERNETICS*.

Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. (n.d.). An algorithm for the traveling salesman problem 1.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems*.

Nicholson, T. A. J. (1971). *Optimization in industry / vol. 1, optimization techniques.*

NVWA. (n.d.). *Gebieden met verbod op gebruik oppervlaktewater | plantenziekten en plagen | nvwa.* Author. Retrieved from `https://www.nvwa.nl/onderwerpen/plantenziekten-en-plagen/bruinrot/verbodsgebieden-gebruik-oppervlaktewater`

Rachmawati, D., Sihombing, P., & Sitorus, V. A. (2020, 12). Capacitated multi depot multi vehicle routing problem using genetic algorithm (case study: watering the medan city park). *Journal of Theoretical and Applied Information Technology*, *98*, 4215-4227.

"Rijksoverheid". ("2023"). *"welke verkeersregels gelden er voor landbouwvoertuigen?".* Retrieved from `"https://www.rijksoverheid.nl/onderwerpen/voertuigen-op-de-weg/vraag-en-antwoord/welke-verkeersregels-gelden-er-voor-landbouwvoertuigen"` ("Accessed: September 15, 2023")

Rio, F., & Harahap, S. (2023). Study vehicle routing problem using nearest neighbor algorithm. *J. Phys*, 12027. doi: 10.1088/1742-6596/2421/1/012027

Steenbreek, S. (2015). *Missie - stichting steenbreek.* Retrieved from `https://steenbreek.nl/focus/`

Toth, P., & Vigo, D. (2014). *Vehicle routing : problems, methods, and applications*.

Urbanization - understanding global change. (n.d.). Retrieved from `\url{https://ugc.berkeley.edu/background-content/urbanization/},year={2023},`

van de Berk. (n.d.). *Watering trees: a practical guide.* Retrieved from `https://www.vdberk.com/advice/watering-trees/`

Wang, M., Ma, T., Li, G., Zhai, X., & Qiao, S. (2020). Ant colony optimization with an improved pheromone model for solving mtsp with capacity and time window constraint. *IEEE Access*, *8*, 106872-106879. doi: 10.1109/ACCESS.2020.3000501

Zare-Reisabadi, E., & Mirmohammadi, S. H. (2015, 12). Site dependent vehicle routing problem with soft time window: Modeling and solution approach. *Computers & Industrial Engineering*, *90*, 177-185. doi: 10.1016/J.CIE.2015.09.002

Zhang, H., Ge, H., Yang, J., & Tong, Y. (2021, 4). Review of vehicle routing problems: Models, classification and solving algorithms. *Archives of Computational Methods in Engineering 2021 29:1*, *29*, 195-221. Retrieved from `https://link.springer.com/article/10.1007/s11831-021-09574-x` doi: 10.1007/S11831-021-09574-X

A APPENDIX

A.1 Appendix A: Taxonomies

This appendix includes the taxonomy used in the literature review of Chapter 3.