ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

MASTER ECONOMETRICS & MANAGEMENT SCIENCE

Specialisation: Quantitative Logistics & Operations Research

Solving the Periodic Vehicle Routing Problem using Ant Colony Optimisation combined with Tabu Search

Author:

Marleen Hielkema

Student number:

620311

Supervisor:

Dr. R. Spliet

Second assessor:

Dr. P.C. Bouman

Abstract

Great industrial producers such as factories produce a continuous amount of waste which needs to be collected on a regular basis due to storage reasons. The exact amount of waste produced by these customers is known and therefore a collection planning can be made over a certain planning horizon. Optimising this planning is known as the Periodic Vehicle Routing Problem (PVRP). The most challenging part of this problem is to assign customers to visiting days, after which the routs on each day are optimised using known algorithms for the Vehicle Routing Problem. In this research, the structure of the PVRP is converted to the that of the Machine Scheduling Problem (MSP) such that the PVRP can be solved using algorithms for the MSP. The solving method investigated in this research is Ant Colony Optimisation algorithm combined with Tabu Search. The combination of the two algorithms knows many different settings and extensions which are all compared to be able to decide on the details of the final algorithm. The combined algorithm improves the initial solution significantly and seems suitable to solve the PVRP, but the computation time is very long for large data sets.

October 27, 2022

Contents

1 Introduction

This research considers the periodic vehicle routing problem (PVRP), which is an extension of the regular vehicle routing problem (VRP). The topic is provided by GreenRoutes, a start-up specialised in creating algorithms that solve routing problems within the great waste industry. This type of waste requires specific vehicles and schedules, thus specific models. A logistic challenge within the routing industry is to create such specific models, since it requires complicated algorithms to solve the problem to optimality. It must be done cost-efficiently, taking into account the customers' requirements as well as capacity and time constraints of the vehicles.

Solving methods for the regular VRP aim to create efficient routes for a set of customers while minimising travel distance or travel time for one time period, for example, a day. The PVRP extends this problem to multiple time periods, referred to as the planning horizon. For the PVRP, an additional piece of information is required from the customer, namely how often a customer wishes their waste to be collected in the planning horizon, referred to as the visiting frequency. For each visiting frequency, there exists a set of feasible visiting schedules which take constraints into account to ensure evenly spread visitations. The biggest distinction between the VRP and the PVRP is that an algorithm to solve regular VRP creates routes, while an algorithm to solve the PVRP assigns customers to visiting schedules and then creates routes on these days. To solve the problem, an algorithm is required which optimises over two aspects.

Since the creation of routes is studied extensively in literature, this research mainly focuses on the assignment of customers to visiting schedules. To be able to use existing literature, the structure of the PVRP is considered as the structure of a Machine Scheduling Problem (MSP). The input of an MSP is a certain amount of jobs which have varying processing times and need to be assigned to a number of identical machines such that the total processing time is minimised. This structure can be used in the PVRP, the jobs are the customers and the machines can be seen as the possible visiting schedules for the customer. When solving the PVRP, we want to find the combinations of customers and visiting schedules which allows for the least total travel time.

The algorithm proposed in this research to find these optimal combinations is a combination of Ant Colony Optimisation (ACO) algorithm and Tabu Search (TS). The ACO assigns a score to the combinations of customers and visiting schedules based on the objective values of the solutions in which they appear. In the next iteration, these scores are used to create a new solution population. To improve this method, a TS heuristic is executed each iteration of the ACO. Furthermore, different options to create the initial solution population are evaluated. Since a solution of the PVRP consists of a routing schedule for a longer period of time, the focus of the solution method does not lay on a short running time, because the problem does not have to be solved regularly.

The combination of the two algorithms gives many different options for the settings and has some possible extensions. The main purpose of this research is to find out whether the combined algorithm is a suitable algorithm for the PVRP. The experiments in this research are designed to investigate the settings for the algorithm, this is done by comparing the quality of the algorithms with all these optional settings. Due to running time reasons, the algorithm is tested on instances which contain a selected amount of data from the original data set.

At first, a literature review on the known solution methods for the VRP and PVRP is provided in Section 2. Secondly, the precise problem description is given in Section 3. Then, methodology on the solving methods researched in this thesis is provided in Section 4. In Section 5, a description of the data used to test the algorithm is given. The results on testing the algorithm for different settings are presented in Section 6 and the results for the different data sets are presented in Section 7. Lastly, Section 8 provides a conclusion which includes the limitations and recommendations for further research.

2 Literature Review

The PVRP introduced in Section 1 consists of two parts; assigning visiting days to customers and creating routes on all the days in the planning horizon. The first part is of the form of a Machine Scheduling Problem (MSP), whereas the second part is a vehicle routing problem (VRP). Both the MSP and the VRP are studied extensively in literature, an overview of relevant literature on these problems is given in this section.

At first, literature on the VRP is presented in Section 2.1. Secondly, in Section 2.2, an overview of literature on the MSP is given.

2.1 Vehicle Routing Problem

The VRP is a generalisation of the Travelling Salesman Problem (TSP) which is elaborated on by Flood (1956). The VRP was first stated as The Truck Dispatching Problem by Dantzig & Ramser (1959) where the difference with the TSP lies within the constraints; in the TSP one salesman may end its route at any customer, while in the VRP the vehicle must end at the same location as it started and more than one truck may be operating. Lenstra & Kan (1981) investigated the computational complexity of the VRP and concluded that the VRP is NP-hard, thus, the problem requires solving heuristics.

A nearest greedy constructive heuristic is created by Mat et al. (2017). For every iteration, all distances to the most recently added customer are calculated and the nearest is selected as the consecutive one. Another option is to create routes by iteratively adding a semi-random customer to a route. This method is proposed by Mancini (2017) and creates routes by assigning probabilities to each customer that has to be placed in a route, these probabilities depend on the travel time to each customer. Solomon (1987) created an algorithm which is referred to as an insertion algorithm, it iteratively adds customers by a certain criterion. The initial route contains the beginning and ending point of the route (the depot), then adds the best customer at the best spot of the route by this criterion. This insertion algorithm seems promising in literature and is easy to implement.

Evolutionary Algorithms often come up in literature when solving the VRP and its extensions, these are heuristic search methods based on Darwinian evolution. Two evolutionary algorithms are interesting in particular, the Genetic Algorithm (GA) and the Ant Colony Optimisation (ACO).

The GA creates a solution population, referred to as chromosomes. Each chromosome is evaluated and given a fitness score, which determines the chances of the chromosome surviving in the next generation. In each iteration, the chromosomes are selected, crossed over and mutated, where the combination of the chromosomes is based on the fitness scores. Thangiah (1995) and Thangiah & Salhi (2001) propose solving methods for the VRP where they use GA to create clusters for the vehicle routes. Then, the routes are post-optimised to make, since the solutions created by the GA are not always feasible. Tan et al. (2000) provide a more extensive GA for the VRP, after which the new solution population is accepted, some solutions are improved. This improvement is called hill-climbing and selects part of the population to perform a delete and best insertion method.

The ACO method assigns fitness values, called pheromone levels, to the arcs of the graph on which the VRP is defined. In each iteration, these pheromone levels are updated based on the travel time between the customers and the quality of the routes in which the arcs are included. Based on these pheromone levels, new routes are created. ACO is studied extensively in literature since the ants creating paths used in the construction phase of the algorithm, can be interpreted as the vehicles driving the routes. Bell & McMullen (2004) provide an ACO making use of the fact that an ant using a short route will return to the depot earlier and marks its path twice before other ants have returned. Li et al. (2009) propose an algorithm in which lower- and upper bounds are assigned to pheromone levels, this avoids getting stuck in a local optimum.

The algorithms stated above know many extensions on in more recent research. Both GA and ACO seem promising in literature, but take many iterations and local search procedures before useful solutions arise.

When a solution is created using ACO, GA, or a constructive heuristic, local search heuristics can be used for improvement. Yu & Yang (2011) propose some 1- and 2-point crossovers on the routes, which are executed after the ACO has terminated. Wang et al. (2020) improved the algorithm of Yu & Yang (2011) by using it to create an initial solution population. On these solutions, six types of local search are applied and Simulated Annealing is used to decide on the acceptance of these solutions.

Since the aim of this research does not lay on the creation of efficient routes, the local search heuristics are not elaborated on further.

2.2 Periodic Vehicle Routing Problem

Two popular approaches to the PVRP will be elaborated on: a two-phase heuristic and to consider the structure of the PVRP as an MSP. At last, local search heuristics are studied which are useful in the second approach.

The two-phase method divides the process in two stages: initialise the solution using a constructive heuristic, then perform local search methods.

Pirkwieser & Raidl (2008) stated a hybrid heuristic which consists of a constructive heuristic and a Variable Neighbourhood Search (VNS) which allows to search through infeasible solutions. To initialise, the customers are ordered based on their angle to the depot, after which routes are created on each day by iteratively allocating the ordered customers to a route. Then for the VNS, three neighbourhood structures are used to improve the results. At last, a 2-opt local search is performed, in which worse solutions are allowed to avoid a local optimum, until no further improvement is found.

Michallet et al. (2014) implemented a Multi Start Iterative Local Search (MSILS), which avoids local optima by using a partly randomised constructive heuristic to create an initial solution population. Then, a local search heuristic is applied to all the solutions. Using this method, the solution space is explored thoroughly. The results of the method are very promising since for small instances it finds all proven optima and for some larger instances, it outperforms existing methods.

The structure of the second method, Ant Colony Optimisation (ACO), is elaborated on in Section 2.1. To use this method on the PVRP, the problem structure is converted to the Machine Scheduling Problem (MSP) as follows; in solving methods for the MSP, jobs are assigned to machines, in those for the PVRP, visiting days are assigned to customers. So, when considering the jobs to be assigned as visiting days, and the machines as customers, the ACO algorithms for the MSP can be converted to the PVRP. Thus, research on ACO for the MSP and PVRP is investigated.

Yu & Yang (2011) created an ACO for the PVRP, where multi-dimensional pheromone levels are set up that contain quality information on both the day and route allocation. When the routes are created, they are improved using 1- and 2-point crossovers. The results are promising, but the computation time for instances containing a long horizon planning and a great number of customers is long. Wang et al. (2020) used the findings of Yu & Yang (2011) to create an initial solution population, after which a score is given to each solution using a multi-objective quality function. Using the scores of all the solutions, simulated annealing is performed to decide on the acceptance of the solutions. At last, local search heuristics are performed. Again, the results seem promising, but the running time is not compatible with the provided data set.

Arnaout et al. (2010) propose an ACO for the MSP where the pheromone levels depend on the job, machine, and time that the job is executed. This can be interpreted as the customer, the schedule and the spot in the route of that day for the PVRP. After the ACO, two simple local search algorithms are performed, one which swaps customers and schedules, and one which assigns new schedules to 5 per cent of the population. Aspects of this paper can be interesting since the results seem promising and details on the functions within the ACO apply to the PVRP. Later on, follow-up research is issued, which focuses on escaping local optima by re-initialising pheromone levels when there is a sign of convergence. (Arnaout et al. 2014)

Arnaout et al. (2008) came up with an algorithm in which ACO is used to assign jobs to machines using one-dimensional pheromone levels which indicate the quality of assigning a certain schedule to a certain customer, after which a form of the TSP applied to the MSP is solved for each machine. Then, local search heuristics are performed to improve the results.

The latter two algorithms seem both promising and compatible with the data set. There are many possibilities to adjust, extend and improve the methods, which makes it interesting to investigate them.

Multiple promising types of local search heuristics for the MSP are found in past literature.

Ying et al. (2012) propose to use Simulated Annealing to improve an initial solution of the MSP. After the initial solution is created, the next solution is selected from a certain neighbourhood. Whether a solution is selected, depends on the difference between its objective value with the current optimal objective value. When the objective value is worse than the current, it is selected by a probability which decreases when the solution gap increases. This is repeated until a stopping criterion is met.

Helal et al. (2006) solve the MSP by first optimising the job sequence, and then assigning these sequences of jobs to machines using Tabu Search. An initial solution is made, after which Tabu Search is performed within the order of the jobs and the assignment of jobs to machines. The Tabu List keeps track of the best sequences that were used on certain machines. The method seems to perform well for large instances. When a perturbation is performed on the current solution, the inverse move can be declared tabu for a specific number of iterations. (Gmira et al. 2021) Since it is computationally expensive to keep track of the exact inverse move, one can decide to make adjustments to a customer tabu.

Udomsakdigool & Kachitvichyanukul (2008) proposes a Multi-Colony ACO which divides the solution population into n colonies, all containing m solutions. Local search heuristics are performed on these solutions, and then from each colony, the best solution is picked to update the pheromone levels.

3 Problem description

Assume the planning horizon consists of D days, then the PVRP is formally defined on the graphs $G_d = (V_d, A_d)$ for $d \in \{1, \ldots, D\}$, where V_d is the set that contains its vertices and A_d is the set that contains its arcs. Vertex $\{v_0\} \in V_d$ represents the depot and vertices $\{v_1, \ldots v_n\} \in V_d$ represent the customers that need to be visited on day d. Each arc $a_{ij} \in A_d$, where $i \neq j$, is assigned a value which represents its cost. This can, for example, be the travel time between customer v_i and v_j.

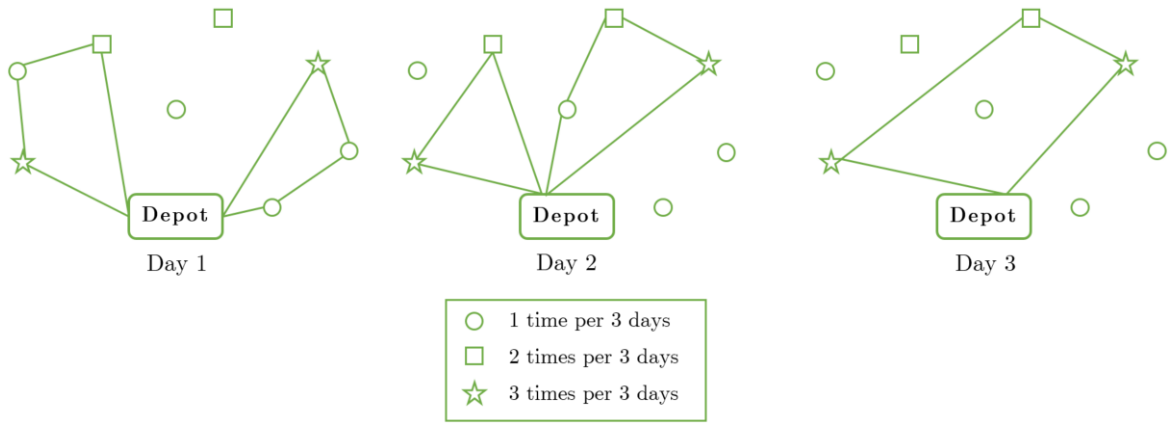
On each day of the planning horizon, the vehicles considered in this research all have the same loading capacity Q and a specific time frame in which they may operate, denoted as $t_0 = [b_0, e_0]$, considered as the working day of the vehicle drivers. Also, the vehicle has to end its route at the same location as it started, which is the depot (v_0). As in the regular VRP, the PVRP requires that all the demand of the customers is met.

Customers need to provide a positive demand, $d_i \in \mathbb{R}$, which in this case is the amount of garbage they want to have picked up in litres. Additionally, the customer must indicate how many times it wants to be visited in the planning horizon, this is the visit frequency $f_{v_i} \in \mathbb{N}$ of customer $v_i \in V_c$, where V_c is the set of all distinct customers in the planning horizon. Each visit frequency f_{v_i} has a set of corresponding visiting schedules, $S_{f_{v_i}}$, where a visiting schedule consists of a set of days on which a customer can be visited. The amount of days between consecutive visits differ per visit frequency, the specific visiting schedules for each f_{v_i} are stated in Section 5.

Extra customer constraints to this problem are the time windows in which it is possible to visit the customers. The time window of customer i will be denoted as $t_i = [b_i, e_i]$, where b_i is the beginning time of the time window and e_i the ending time. A vehicle is allowed to arrive too early and wait, but it is not allowed to arrive later than e_i. Each customer $v_i \in V_c$ has a service time s_i, which indicates how long the collection of its waste takes.

The objective is to minimise the total travel time of the period to serve all customers. The aim is to find an optimal and feasible solution such that the constraints on time frames of the vehicles and customers and the visit frequency of the customer are met, next to the regular constraints of the capacitated vehicle routing problem. The solution assigns each customer to a visiting schedule and to a spot in a route. The desired output consists of vehicle routes on each day of the planning horizon. As an example, an output of a planning horizon of three days, where customers $v_i \in \{1, 2, \ldots, 8\}$ have visit frequency $f_{v_i} \in \{1, 2, 3\}$ is illustrated in Figure 1.

Figure 1: Example of a solution of the PVRPTW on a planning horizon of three days



As mentioned in Section 1, the problem of assigning schedules to customers has the same structure as the Machine Scheduling Problem (MSP), since the MSP entails assigning jobs to machines. When customers are assigned to visiting schedules, the objective value of the period is evaluated by calculating the total travel time of the period. This is done by solving the VRP on each day of the planning horizon, and adding the travel times of all routes on each day.

4 Methodology

In this research, a combination of ACO and TS is proposed to solve the PVRP. In this section, detailed information is provided on both algorithms and an elaboration is given on the possibilities of combining the two.

At first, the ACO method is discussed in detail in Section 4.1. Secondly, an elaboration on the local search method, Tabu Search, that will be used in the algorithm is provided in Section 4.2. Then, an explanation on the algorithms to initialise the solution population is given in 4.3. Section 6.3 presents the possible combinations of the methods. Finally, Section 4.5 describes the method used to create routes on the days of the planning horizon once the customers are assigned to visiting schedules in each iteration.

4.1 Ant Colony Optimisation

ACO is an evolutionary algorithm which is used to solve problems that can be defined on graphs. The algorithm simulates the pheromone-based communication of real-life ants. During the algorithm, artificial ‘ants’ move through the solution space of the problem. The arcs that appear in an ant are given a fitness score, ‘pheromone level’, based on the objective value of the ant. The pheromone levels of all the ants are combined and converted to probabilities which are used to create the next solution population. Thus, when an ant represents a high-quality solution, the chance of the arcs of this ant reappearing in the next solution population is higher than when the ant is of poor quality.

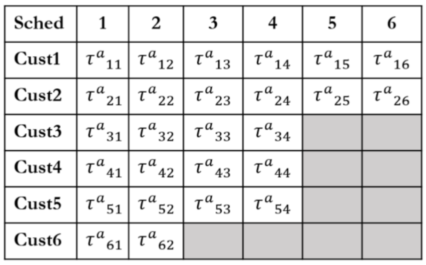
As mentioned in Section 1, the PVRP has a structure like the MSP. In literature on the ACO for the MSP, the ‘ants’ in the ACO represent the solutions which consist of combinations of certain jobs and machines. In the case of the PVRP, the ants consist of combinations of customers and visiting schedules. The pheromone levels of the combinations are based on the objective values of the solutions they appear in, which in this case is the total travel time.

As mentioned earlier in this section, the ACO is typically used to solve problems which can be represented on a graph. The MSP and PVRP are not representable on a logical graph but can be defined easily in a table which contains the pheromone levels of all the combinations appearing in the solution space. This indicates the ACO is suitable for solving the PVRP.

At first, an elaboration on the calculation of the pheromone levels is given in Section 4.1.1. Then, the conversion to the probabilities is discussed in Section 4.1.2.

4.1.1 Pheromone levels

The solution population consists of multiple ants, each ant has an objective value which is used to calculate the pheromone levels of the combinations of all customers and schedules that appear in that ant. Then, the master pheromone level matrix is calculated, this master pheromone level matrix contains the combined information of all the ants that appear in the population. The equations to calculate the pheromone levels are inspired by Yu



Sched	1	2	3	4	5	6
Cust1	$\tau^a{}_{11}$	$\tau^a{}_{12}$	$\tau^a{}_{13}$	$\tau^a{}_{14}$	$\tau^a{}_{15}$	$\tau^a{}_{16}$
Cust2	$\tau^a{}_{21}$	$\tau^a{}_{22}$	$\tau^a{}_{23}$	$\tau^a{}_{24}$	$\tau^a{}_{25}$	$\tau^a{}_{26}$
Cust3	$\tau^a{}_{31}$	$\tau^a{}_{32}$	$\tau^a{}_{33}$	$\tau^a{}_{34}$		
Cust4	$\tau^a{}_{41}$	$\tau^a{}_{42}$	$\tau^a{}_{43}$	$\tau^a{}_{44}$		
Cust5	$\tau^a{}_{51}$	$\tau^a{}_{52}$	$\tau^a{}_{53}$	$\tau^a{}_{54}$		
Cust6	$\tau^a{}_{61}$	$\tau^a{}_{62}$				

Figure 2: Pheromone matrix example

& Yang (2011) and are adjusted to the problem stated in this thesis.

An example of the pheromone level matrix with 6 possible schedules and 6 customers is pictured in Figure 2. Customers 1 to 3 have six possible visiting schedules, customers 3 to 5 have four possible visiting schedules and customer 6 has two possible visiting schedules. The probability matrix has the exact same structure.

The formula to calculate the individual pheromone level of the combination of customer c and schedule s (hereafter referred to as $c - s$) for an ant a is as follows

$$\tau_{cs}^{a} = 1 - \frac{|L^{\text{opt}} - L^{a}|}{L^{\text{opt}}}$$

Where L^{opt} is the best objective value found so far, and L^{a} is the objective value of the ant.

Once the individual pheromone levels of the $c - s$ appearing in the solution population are calculated using the formula above, the pheromone levels of the ants are combined in the master pheromone matrix such that the next solution population can be created based on all the information retrieved from the current solution population. The following formula calculates the master pheromone level for all possible $c - s$

$$\mathrm{T}_{cs}^{\text{new}} = (1 - \rho) \times \mathrm{T}_{cs}^{\text{old}} + \sum_{a \in A(c-s)} \tau_{cs}^{a}$$

Where $\mathrm{T}_{cs}^{\text{old}}$ is the pheromone level before updating and ρ represents the evaporation rate. The evaporation rate is an important parameter in the ACO, it indicates the weight of the information from the previous iteration. The aim of the evaporation rate is to combine the old and the new information in such a way that old information is not lost, but the ants do not end up in a local optimum. Furthermore, $A(c - s)$ is the set of all solutions in the population containing the combination $c - s$.

4.1.2 Probabilities

Once the pheromone levels are calculated, they are converted into probabilities which indicate the chance of combination $c - s$ to appear in an ant when constructing the next solution population. Again, the formulas used are based on Yu & Yang (2011).

The chance of combination $c - s$ to be chosen when constructing a new ant, p_{cs}, is calculated as follows

$$p_{cs} = \frac{(\mathrm{T}_{cs})^{\alpha} \times (\eta_{cs})^{\beta}}{\sum_{s} (\mathrm{T}_{cs})^{\alpha} \times (\eta_{cs})^{\beta}}$$

Where T_{cs} represents the pheromone level, α and β are the weight parameters which indicate the relative influence of the pheromone levels and η_{cs} is the visibility of the arc denoting the quality of $c - s$ overall. Before elaborating on the visibility, an important measure is highlighted: the time-customer ratio (tc-ratio) which indicates the quality of a single route and is calculated as follows

$$tc - \text{ratio} = \frac{\text{amount of seconds the route takes}}{\text{amount of customers served in the route}}$$

When a route takes a long time and serves just a few customers, the ratio is high, indicating the route is 'bad'. This measure can be used to calculate the visibility of combinations appearing in a route.

Visibility

The visibility η_{cs} of $c - s$ is an extra quality measure and indicates how 'visible' the combination is when calculating the probability. The use of visibilities allows one to decrease or increase the probability of multiple $c - s$ at once, instead of looking at $c - s$ individually. The matrix which contains the visibilities has the same structure as the pheromone and probability matrix, depicted in Figure 2. In ACO algorithms for the VRP, the visibility of an arc between two customers is calculated by taking the inverse of the travel time between the two customers. (Bell & McMullen 2004) For the MSP, an obvious option is to use the inverse of the processing time of a certain job on a specific machine. (Arnaout et al. 2010)

For the PVRP, the quality of a combination can not be defined beforehand but is calculated each iteration. This research will investigate three options for visibility updating:

1. **Worst route.** For each customer, the route in the solution population with the lowest quality in which the customer is included is detected. The visiting schedule for the customer corresponding to this solution is given a visibility of $\eta_{cs} = \frac{Q}{\text{max\_ratio}_{cs}}$, where max_ratio$_{cs}$ is the ratio value of the worst route found, and Q is to be decided on beforehand. The other possible schedules of these customers are assigned an initial visibility value.

2. **High quality route.** To preserve high-quality routes in a solution, one can increase the chance of the appearance of these routes in the next solution population. This can be done by increasing the visibility of all combinations that appear in these routes simultaneously. To achieve this, R good routes are detected by calculating their tc-ratio and the visibilities of all the $c - s$ in these routes are multiplied by a factor q for which the value is to be decided on. This method is inspired by the method created by Helal et al. (2006).

3. **Tabu.** To avoid a local optimum, a visibility can be calculated that decreases the more $c - s$ appears in the solution population. At first, the amount of times all the $c - s$ appear in the solution

population is evaluated (appearance$_{cs}$), after which the following formula is used to calculate the visibility value of $c - s$

$$\eta_{cs} = \frac{n_{\text{iterations}} - \text{appearance}_{cs}}{n_{\text{iterations}}}$$

Where $n_{\text{iterations}}$ represents the number of ACO iterations.

This visibility includes a type of TS in the algorithm by disfavouring combinations which have appeared in solutions of the previous iteration.

Fixating routes

An alternative method to keep high-quality routes in the solution population is to fixate the combinations which appear in routes in the solution population which seem most promising. After solution construction and pheromone, probability and visibility updating, it is possible to adjust the probability matrix again to completely fixate the best route(s) that are found in the solution population. This is done by finding the best route(s), and then setting the probabilities of all $c - s$ combinations that appear in it equal to 100% in the probability matrix. High-quality routes are the routes that have a low $tc-$ratio.

The routes to fixate are extracted from different solutions, so it is important to check whether a customer appears in more than one selected route since it is not possible to fixate a customer with multiple different visiting schedules. To prevent fixating the same route each iteration, a tabu list is kept containing customers that are fixated in previous iterations. The length of this tabu list differs per data set size since the number of routes in the solution depends on this. When a small number of routes appear in a solution, one wishes to fixate a small number of routes since the solution may converge to a local optimum otherwise. When a large number of routes appear in a solution, it may be interesting to fixate more routes such that the algorithm gives higher-quality solutions in a shorter amount of time.

This research will investigate which settings of parameters for the formulas elaborated on above are suitable for the PVRP.

4.2 Tabu Search

To improve the current solution population, a local search algorithm is performed on (part of) the ants in the solution population. The local search choice is Tabu Search (TS), which is a degradation local search. TS forces the solution to escape from a local optimum by allowing worse solutions to be explored. First, a solution neighbourhood is created for the solution to be investigated. Then, a tabu list is created to which routes, or parts of routes, which may not appear in the next solution are added each iteration. This list makes sure that solutions that do not seem very promising are explored, which can avoid getting stuck in a local optimum. In each iteration, the best solution in the neighbourhood which is not (partly) tabu is chosen as the next solution and the tabu list is updated. This is repeated until a stopping criterion is met.

Neighbourhoods and Tabu List

The neighbourhood of a solution a is a set of solutions, $n(a)$, which contains modified versions of a. The tabu list contains a list of solutions, or parts of solutions, which may not be selected in the solution of the next iteration of TS. When a new solution a' is chosen from the neighbourhood $N(a)$, one needs to check whether it is not tabu. If it is tabu, another solution from the neighbourhood must be chosen. The setting of the length of the tabu list is important, if it is too long, one might avoid good solutions, but when it is too short, one might get stuck in a local optimum.

Since the structure of the PVRP is comparable to the structure of the MSP, it is interesting to apply sort-like neighbourhoods to our problem. The following neighbourhoods and tabu list types are used in this research to explore perturbations of ant a

- **Best insertion** (n_{ins}) – A visiting schedule is removed from the customer and the visiting schedule resulting in the best objective is given. The tabu list keeps track of customers whose schedules are modified. (Cordeau et al. 2001)
- **Highest ratio** (n_{ratio}) – Find the route with the highest tc-ratio and give all the customers a different schedule. The tabu list contains the exact routes which have been redistributed. This neighbourhood is inspired by a TS proposed by Helal et al. (2006) for the MSP, where the busiest machines have their jobs reassigned.
- **Swap** (n_{swap}) – Choose a random customer and find the customer with the same visiting frequency and a different visiting schedule which improves the objective value the most when their visiting schedules are swapped. The tabu list consists of customers who have had their schedules swapped.

As mentioned, the tabu list length is to be decided on, this differs per neighbourhood and what aspect of the solution is added to the tabu list. When customers are made tabu, implying that they cannot be adopted for a certain amount of iterations, the tabu list can be relatively long since the data set includes many customers. When complete routes are made tabu, it is not wished to allow the length of the tabu list to be very long. First of all, the chances of creating the exact same route are not that big, so the tabu list has a less impactful function. Secondly, when a route has a very low tc-ratio and reappears in solutions of the next iterations, it can have a relatively large impact on the objective value compared to a single customer that has its schedule swapped. Also, the longer the length of the tabu list is, the more computation time it takes to check whether a solution is tabu.

Combining neighbourhoods

To search a wider part of the solution space, it is interesting to investigate the possibility of using multiple different neighbourhoods while performing Tabu Search. This implies that different neighbourhoods are used for a certain amount of iterations to find the next solution. There are a lot of options when executing Tabu Search using different neighbourhoods; the amount of iterations for each neighbourhood and the order of these neighbourhoods.

4.3 Boosting the initial solution population

When creating a solution population to start the ACO with, it is important to take two aspects into account; the diversity and the quality of the solution population. The presence of diversity in the initial solution population is important to avoid a local optimum. To prevent the need for too many iterations in the ACO, the quality of the initial solution population is also important. It is essential to balance these two conditions for a valuable initial solution population.

The initial solution population is made using the initial probability matrix which contains equal probabilities for the possible schedules for each customer. For example, if a customer has 5 possible feasible schedules, all these schedules have a probability of 20%. After the initialisation, the population is boosted before it is improved by the ACO. Three different boosting algorithms are implemented; Multi-Colony, Tabu Search boosting, and Simulated Annealing election.

1) Multi-Colony

Udomsakdigool & Kachitvichyanukul (2008) propose an algorithm in which the solution population is divided into colonies which are boosted in various ways to create a diverse population.

After initialising the probability matrix, $C \times A$ ants are created. This group of solutions is separated into C colonies which all contain A ants. Then, each colony uses a different neighbourhood (n_i for $i \in \{1, \dots, C\}$) to execute TS, this ensures diversity in the complete solution population. From each colony, the best few solutions are used to update the pheromone levels.

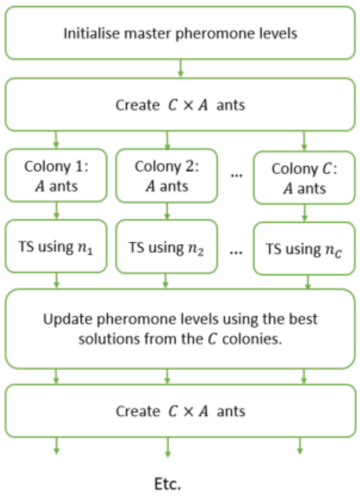


Figure 3: Multi-Colony method

Figure 3 illustrates the algorithm structure.

2) Tabu search boost

This method boosts the initial solution population by performing TS on the ants before using them to update the pheromone levels. To maintain variety in the solution population one does not want to apply TS to all the solutions, thus a proportion of the ants (p_{TS}) is chosen to perform TS on. Another way to maintain variety is to choose the amount of TS iterations randomly from a certain range.

3) Simulated Annealing election

This method is initialised by creating a solution population of $2 \times A$ ants, after which TS is performed on all the solutions. Then, using a Simulated Annealing criterion, A ants are selected to form the next population. The probability that ant $a \in A$ is chosen from the population is calculated as follows

$$\begin{aligned} p_a &= \exp\left(\frac{-|L^a - L^{\text{best}}|}{T}\right) \\ T &= L^{\text{worst}}/v \\ v &= \frac{\ln(p_{\text{worst}}) \times L^{\text{worst}}}{-|L^{\text{best}} - L^{\text{worst}}|} \end{aligned}$$

Where L^a is the objective value of ant $a \in A$, L^{worst} is the objective value of the worst solution, and L^{best} is the objective value of the best solution in the solution population. Here, $p_{\text{worst}} \in [0, 1]$ represents the chance of the worst solution in the solution population to be picked and must be decided on beforehand. Setting a higher value of p_{worst} may result in a solution population with a wider range of objective values, where a lower value results in solutions of higher quality.

Looping over the $2 \times A$ constructed ants, a random number $r \in [0, 1]$ is chosen and when $r < p_a$, ant a is added to the solution population, otherwise it is skipped. This is done until A ants are chosen.

The method is based on the findings by Yu & Yang (2011) and is adjusted to be useful for the initialisation of the PVRP ACO.

4.4 ACO and TS

In literature, the combination of ACO and TS is popular for the VRP but has, to our knowledge, not been proposed for the PVRP. Since literature states that both methods are promising for the MSP, and ACO is often combined with a local search, this research will investigate such combined solution methods for the PVRP. There are two interesting ways to include TS in ACO; perform TS each ACO iteration or use a tabu structure when updating the visibilities.

1) TS on solution population

The first option is to perform TS on all the ants of the solution population in each ACO iteration. After constructing and evaluating the initial solution population, the combinations appearing in the ants are provided with pheromone levels and probabilities. Using these probabilities, a solution population is constructed after which a certain amount of TS iterations is performed on the ants. Due to TS, the solutions are of higher quality when they are used to update the pheromone levels for the next ACO iteration.

2) TS in updating visibilities

The second option is to implement the TS in the probability level updating phase of ACO, which is done by updating the visibility matrix according to visibility option 'Tabu' from Section 4.1.2.

Both options can be extended in multiple ways. First of all, one can boost the initial population before starting ACO using the Multi-Colony, Tabu Search boost or Simulated Annealing election algorithms elaborated on in Section 4.3. Furthermore, it is possible to fixate the best routes that are present in the solution population as explained in Section 4.1.2. The visibility and pheromone matrices can be updated using different parameter settings, elaborated on in Section 4.1.1 and 4.1.2.

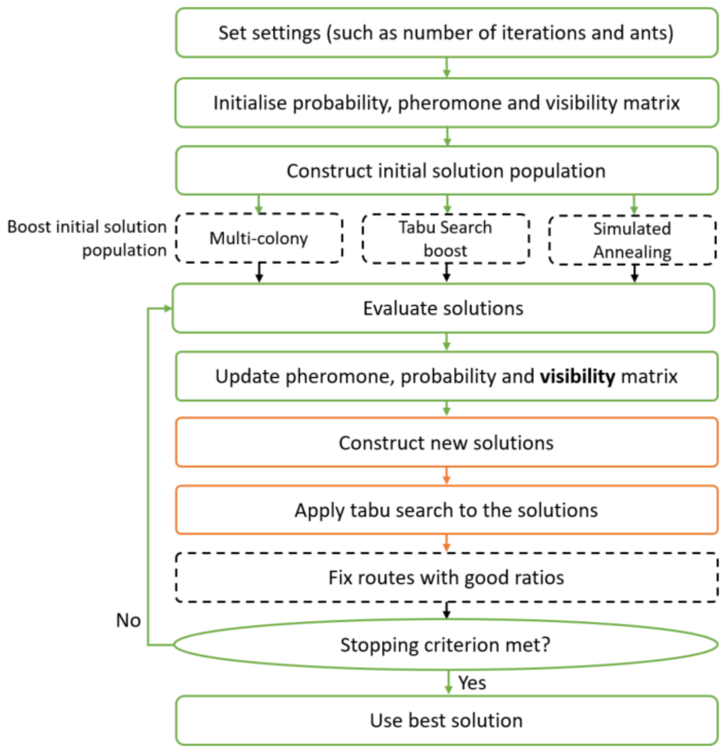
The two options are illustrated in the flowchart in Figure 4. The black dotted cells represent the optional features, these are considered as extensions. The orange cells are steps that belong to option 1, they are not included in the second option. It is important to note that the biggest difference between options 1 and 2 is that the visibility matrix is updated differently and that the TS is not performed each iteration in option 2.

This research will focus on the combination of ACO and TS, where it is important to find out which of the two options and which extensions have the most potential on succeeding to find a good solution for the problem in a reasonable amount of time.

4.5 Constructive Heuristic for routes

In each iteration of the ACO, the objective values of the solutions are evaluated such that the pheromone levels can be evaluated. The objective value of a solution is the total travel time of the period. The solutions created by the ACO and the TS consist of all customers and their visiting schedules, which indicates that the customers that need to be visited on each day are known. To obtain the objective value of the solution, efficient routes need to be created on each day of the planning horizon such that the travel time can be calculated. The problem that needs to be solved for each day is the Capacitated

Figure 4: ACO combined with TS


Set settings (such as number of iterations and ants)
Initialise probability, pheromone and visibility matrix
Construct initial solution population
Boost initial solution population
Multi-colony
Tabu Search boost
Simulated Annealing
Evaluate solutions
Update pheromone, probability and **visibility** matrix
Construct new solutions
Apply tabu search to the solutions
Fix routes with good ratios
No
Stopping criterion met?
Yes
Use best solution

Vehicle Problem with Time Windows (CVRPTW), which implies that for each route the capacity of a truck may not be exceeded, the truck visit must be within a certain time window and the route must start and end at the depot.

The most promising constructive heuristic for the CVRPTW found in literature is the insertion method proposed by Solomon (1987). This method iteratively adds unvisited customers to a route. This is done by looping over all unvisited customers and their possible insertion places while finding the best combination.

After initialising the current route by (d, d), where d is the depot, customers are added using a criterion, $c(i, u, j)$, which indicates the quality of placing customer u between two adjacent customers i and j. The criterion is defined as $c(i, u, j) = t_{iu} + t_{uj} - t_{ij}$, where t_{ij} is the travel time from customer i to j. For each unvisited customer u, the best feasible insertion place is found by calculating $c(i, u, j)$

$$c(i(u), u, j(u)) = \min[c(i_{p-1}, u, i_p)], \qquad p = 1, \ldots, m$$

Where m is the current length of the route. Then, from these unvisited customers and their optimal feasible insertion place, the one is selected which causes the least increasing travel time of the route. This is repeated until all customers are visited, or until the capacity of the vehicle is met.

5 Data

The data provided by GreenRoutes consists of information on 1934 customers. For each customer, data is provided on the time frame in which the customer can be visited, the wished visiting frequency, the amount of waste to be picked up on each visit, the location of a customer in coordinates, and the service time.

The time frame in which customer $i \in V$ can be visited is denoted as $t_i = [b_i, e_i]$. Where both b_i and e_i are time indications such as 11:00, 12:00, etc. Most of the customers have a service time range of [7:00, 15:00]. In the original data set, 283 customers have a time range starting after 7:00 and 16 ending before 15:00. This indicates that the time range constraint will not be very restrictive when solving the PVRP for the instance given.

The solution that is going to be created is for a planning horizon of one month, which contains 20 working days. These are four consecutive weeks in which the vehicles will operate on Monday, Tuesday, Wednesday, Thursday, and Friday. The visiting frequency of customer i is defined as $f_i \in \{1, 2, 4, 8, 12, 16, 20\}$, indicating the number of times the customer wants its waste to be collected. We want to make sure that the visits scheduled are spread evenly over the 20 days, therefore some restrictions on the visiting schedules are made for all the possible values of the visiting frequency. These restrictions and the number of customers with the respective frequency in the original data set can be found in Table 1.

f_i	**Restrictions**	**No. customers**
1	One of the 20 working days.	399
2	Visit in week 1 and 3, or week 2 and 4. At least two weeks between visits.	328
4	Visited on the same day each week.	843
8	Visited two times per week. At least two days between consecutive visits.	302
12	Visit three times per week. No visits on two days per week.	44
16	Visit four times per week. No visits on one day per week.	2
20	Visit every day on the planning horizon.	16

Table 1: Restrictions and amount of customers per possible visiting frequency

These restrictions result in visiting schedules for each visiting frequency. A visiting schedule of customer i is a set of days on which the customer is visited. A customer has multiple feasible visiting schedules for which the constraint holds, these schedules are all contained in the set S_{f_i}. As an example, if a customer has a visiting frequency equal to 4, it is visited once a week on the same day. Thus, the set S_4 is defined as follows $\{\{1, 6, 11, 16\}, \{2, 7, 12, 17\}, \{3, 8, 13, 18\}, \{4, 9, 14, 19\}, \{5, 10, 15, 20\}\}$. The visiting schedule sets for all the possible visiting frequencies can be found in Appendix A.

The amount of waste to be collected from each customer is the same for each visit that is scheduled in the planning horizon. It is given in litres and ranges from 60 to 10,000 with a mean of 1010 and a standard deviation of 933.

The address and location of the customers is given in latitude and longitude and ranges from [51.7688234, 4.0882661] to [52.1545541, 4.8807865], which is an area of 50 by 50 kilometres. A map on which all customers are pinned can be found in Appendix B. Using the longitude and latitude, a travel time matrix is created using Open Source Routing Machine (OSRM). OSRM is a routing engine which finds the shortest path between two coordinates using real-life travel time information. The travel time matrix contains in column i on row j the travel time from customer i to customer j. It is important to note that the travel time is not symmetric, since the route from customer i to j and from j to i might not be identical.

The time it takes to collect the waste of a customer is set to 90 seconds for each customer.

All vehicles have a capacity of 150,000 litres and operate between 7:00 and 15:00 on working days (Monday, Tuesday, Wednesday, Thursday, Friday). Furthermore, the vehicles must start and end at the same depot.

5.1 Test instances

GreenRoutes provided one data set containing 1934 customers, which is a large amount for the algorithms that are being used in this research. Therefore, smaller instances are created from this data set to test and evaluate the algorithms. Six instances are created by drawing 400 random customers from the original data set, these six instances will be referred to as INS_6. Additionally, this will prevent overfitting the algorithm, which might happen when only one instance is used to tune all parameters in an algorithm.

5.2 Urban area

A large part of the customers contained in the data set provided by GreenRoutes are clustered in industrial areas, since the customers are mostly factories or other industrial companies which produce great waste. The other part of the customers are spread over the entire city Rotterdam. To test the algorithm on a smaller rural area, a data set is created which contains 400 addresses located in Delfgauw, a village near Delft. The latitude and longitude ranges from [51.9841285, 4.388539] to [52.0202618, 4.4156712], and strains an area of 4.3 by 2.5 kilometres. This instance will be referred to as INS_U.

This data set is originally used to test algorithms which optimise picking up private waste. The demand of the customers is adjusted such that the data set simulates the industry customers. Also, visiting frequencies are added, which are simulated such that the proportions match the original data set. A map containing all customers included in this data set can be found in Appendix C.

5.3 Planning horizon

To test the computation time and quality of the algorithms for different sizes of planning horizons, a data set of 400 customers is made for a planning horizon of both 10 and for 30 days, referred to as INS_10d and INS_30d respectively. The visiting schedules are adjusted such that they are in proportion to the planning horizon. Specifications on the adjustment of the schedules can be found in Appendix A.

5.4 Visiting frequencies

The running time and quality of the algorithms are expected to depend on the visiting frequencies of the customers due to flexibility of the placement of customers and the available choices of schedules. Furthermore, when more customers have a high visiting frequency, more customers need to be visited each day, and more routes need to be constructed.

To test the algorithms on data sets with a different frequency composition, three data sets are created. One containing customers with equally distributed frequencies, one containing exclusively low frequencies ($f_i = \{1, 2, 4\}$) and exclusively high ($f_i = \{8, 12, 16, 20\}$) are generated, referred to as INS_eq, INS_high and INS_low respectively.

To summarise, the settings of the algorithm are tested by running all the instances of INS6 and evaluate the average outcome. When the settings are decided on, the performance of the algorithm is tested on the instances INS_U, INS_10d, INS_30d, INS_eq, INS_high and INS_low in Section 7 to investigate whether the algorithm is suitable for instances with different characterisations.

6 Determining settings ACO and TS

Figure 4 visualises the different stages of the algorithm to be investigated in this research. Every stage in the algorithm requires different aspects to test. For both the ACO and TS, the basic settings must be determined before the extensions can be evaluated.

For the ACO, the basic settings include the evaporation rate, the number of iterations, the number of ants and the visibility updating method, these will be investigated in Section 6.1. Then, the extension ‘fixating routes’ is tested in Section 6.1.1. For the TS, the basic settings are the number of iterations and the tabu list length for each neighbourhood, these experiments are elaborated on in Section 6.2. As an extension, the option of testing different neighbourhoods consecutively is tested in Section 6.2.1. Once the settings of the ACO and TS are decided on, the combination of performing TS in each ACO iteration is studied in Section 6.3. At last, the different methods for creating and boosting the initial solution population are tested in Section 6.4.

All experiments are executed on the same laptop which has an Intel(R) Core(TM) i5-7200U CPU, 2.50GHz, 8.00 GB RAM, 64-bit operating system and x64 processor, running Windows 10. All methods are coded in Python and run in PyCharm Professional 2021.3.3 or Spyder IDE 5.1.5.

6.1 ACO

For the ACO, the evaporation rate, the number of iterations, the number of ants and the visibility updating method must be determined. The number of iterations is important to tune to create a balance between the running time and the quality of the solution. Due to the many possibilities of combinations between customers and visiting schedules, it is essential to test the number of ants needed to create an initial solution from which useful information can be retrieved. A concise overview of the algorithm structure is illustrated in Appendix D.

The algorithm is tested for $n_{ants} = \{5, 10, 30\}$, inspired by (Arnaout et al. 2010). The parameters α and β are set to 1 (Arnaout et al. 2008) and the evaporation rate, ρ, is tested for the values 0.01, 0.3 and 0.5 (Arnaout et al. 2010). The number of iterations is set to infinity and a stopping criterion is implemented to find the optimal amount of iterations; When the algorithm does not provide a better solution than the current best for ten iterations, the algorithm terminates.

To be able to compare the performance of the algorithm for the different settings, the improvement of the solution is quantified, which is defined as follows:

$$\text{Improvement solution} = \left(\frac{L^{\text{initial}} - L^{\text{final}}}{L^{\text{initial}}} \right) \times 100$$

Where L^{initial} is the objective value of the best solution in the initial solution population, and L^{best} is the objective value of the best solution found while running the algorithm.

The algorithm contains a random element since the solutions are created based on probabilities. Therefore, it is important to check whether the performance is stable to decide on the design of the experiment.

The ACO can not be tested without the visibilities, since then the solution will not converge and no useful results will appear. Thus, the stability of the algorithm is tested using the visibilities elaborated on in Section 4.1.2 while experimenting on the basic settings.

To check the stability of the performance of the ACO using the different visibilities ‘worst route’, ‘high quality’ and ‘tabu’, the first instance of INS6 is run 8 times and the results are evaluated. The settings for testing the stability of the algorithm are $n_{\text{ants}} = 30$ and $\rho = 0.5$, and for the number of iterations the stopping criterion above is used. The results can be found in Table 2. The goal is to decide on the stability of the performance of the algorithm. Also, this part gives a first impression of the performance of the visibility updating possibilities.

The individual settings for the visibilities are as follows; For the worst route, the settings for testing the visibilities are $Q = 100$, for high quality visibility these are $q = 2$ and $R = 5$, and for the TS visibility there are no settings to decide on beforehand. The definitions of these parameters are elaborated on in Section 4.1.2. The results are reported in Table 2. The standard deviation (STD) is calculated for each of the aspects, which quantifies the stability.

Visibility updating type	**Worst route**		**High quality route**		**Tabu**	
	Avg	STD	Avg	STD	Avg	STD
Improvement $(\%)$	19.76	1.58	2.11	1.03	0.69	0.35
Iterations until termination	43.13	10.03	19.75	5.73	16.63	6.25
Running time (m)	115.63	33.02	36.66	13.13	17.24	7.04
Running time per iteration (m)	2.75	0.78	1.83	0.27	1.04	0.14

Table 2: Stability of ACO with different visibility options

For the ‘worst route’ option, the average improvement is 19.76% with a standard deviation of 1.58% over all the runs, this indicates that the algorithm performs stable and seems promising. On the other hand, for the ‘high quality’ visibility update, the average improvement is 2.10% with a standard deviation of 0.96%, and for the ‘tabu’ visibility update, the average improvement is 0.68% with a standard deviation of 0.34%. These algorithms perform stable but do not seem useful when solving this problem. Different settings may result in a better outcome, but will never outperform the option ‘worst route’, which thereby determines the choice of using the ‘worst route’ visibility for the remainder of the experiments.

Now that the stability and the visibility updating choice are decided on, the next step is to carry out experiments which decide on the number of ants and the evaporation rate for the ACO. Since the algorithm performs stable, the algorithm is tested on all instances of INS6. The average values for iterations until termination, improvement of the solution and running time can be found in Table 3.

When analysing the results in Table 3, it is important to take into account multiple aspects. Note that the algorithm might be run for many iterations, thus it is important to make choices which balance

Ants	5			10			30		
ρ	0.01	0.3	0.5	0.01	0.3	0.5	0.01	0.3	0.5
Improvement of solution (%)	3.23	15.99	13.35	4.36	16.69	14.71	6.91	17.53	19.89
Iterations until termination	19.45	65.40	44.63	33.40	66.83	43.33	38.67	65.67	77.00
Running time (m)	6.99	31.23	16.55	18.17	55.67	35.59	98.31	143.91	235.98
Running time per iteration (m)	0.31	0.44	0.53	0.55	0.83	0.82	2.58	2.22	2.99

Table 3: Results ACO with different settings for ρ and number of ants

the running time and improvement. The settings resulting in the most improvement of the solution are ($n_{\text{ants}} = 30, \rho = 0.5$), but take a lot of computation time. The settings ($n_{\text{ants}} = 5, \rho = 0.3$) are favoured since they provide a comparable outcome in a much shorter amount of time.

The number of iterations for the ACO is not easy to decide on. The ACO for this problem is expensive in running time since it creates 5 solutions which all consist of 20 days on which routes are constructed every iteration. In Table 3 one can find that the average number of iterations for the settings is 65.40, which is very high considering the running time per iteration is 0.44. It is obvious that the more iterations are performed, the better the solution will be. But to avoid long running times while testing the algorithm further, the number of ACO iterations is set to 30. This can be done since the goal of the experiments is to compare the algorithms with different settings.

6.1.1 Fixating routes

Now that the settings for the choice of visibility, evaporation rate, number of iterations and number of ants are decided on, the extension 'fixating routes', which is elaborated on in Section 4.1.2, can be evaluated. At the end of each ACO iteration, after the pheromone levels, visibilities and probabilities are updated, the method is performed. It fixates the combinations of customers which are included in high quality routes. Since the data set is small, one route is fixated per iteration and a tabu list of length 150 customers is kept.

Since we want to compare the algorithm including 'fixating routes' to the algorithm excluding 'fixating routes', both will be run on all instances of INS6 with the settings $n_{\text{ants}} = 5$, $\rho = 0.3$ and $n_{\text{iterations}} = 30$. The results are reported in Table 4.

Fixated routes per iteration	**0**	**1**
Improvement of solution (%)	13.21	14.27
Running time (m)	35.87	18.62
Running time per iteration (m)	1.20	0.62

Table 4: Results fixating routes extension

To conclude, the ‘fixating routes’ method achieves approximately the same results in less running time. The decreased running time is due to the fact that there are fewer routes to construct since the best route is used in all the ants in the next iteration. Therefore, it is useful to include this method in the algorithm.

6.2 TS

The quality of the different neighbourhoods, the respective tabu list length, and the number of iterations for the Tabu Search need to be decided on. The quality of the neighbourhoods indicates which neighbourhoods are useful for the final algorithm. The tabu list and number of iterations are important since they can affect both the quality and the running time of the algorithm. A concise overview of the algorithm structure is illustrated in Appendix D.

As for the ACO, it is crucial to check whether the algorithm performs stable to decide on the design of the experiment of the settings mentioned. The algorithm is tested 8 times with the first instance of INS6. Table 5 contains the results for this stability check, which include the STD of the outcomes of all the runs. This stability check also gives an inside on the performance of the different neighbourhoods.

For the stability check of the neighbourhoods n_{ins}, n_{ratio} and n_{swap}, the allowed length of the tabu list is set to 30, 10 and 30 respectively. The amount of iterations is set to 100 from which the best objective is reported in Table 5. When analysing the standard deviations of the improvement of the algorithm denoted in Table 5, one can conclude that the algorithm performs stable.

Neighbourhood	n_{ins}		n_{ratio}		n_{swap}	
	Avg	STD	Avg	STD	Avg	STD
Improvement $(\%)$	5.99	1.44	12.85	1.40	14.64	1.79
Running time (m)	93.52	48.44	7.47	0.50	158.59	35.96
Running time per iteration (m)	0.94	0.48	0.07	0.01	1.59	0.36

Table 5: Stability check of the TS algorithm for the different neighbourhoods

Now that we find that the algorithm performs stable, the settings for the length of the tabu list for each neighbourhood can be tested. The TS settings will be tested on all of the six instances in INS6 and the average of the improvement on all these instances will be evaluated. We set the number of iterations for each neighbourhood to 30, and evaluate which settings result in the best solution. In Table 6 the results of the experiment are reported.

The first thing that stands out is that for each neighbourhood the different settings for the lengths for the tabu list do not result in significant differences in the improvement, but they do in the running time (per iteration). The second thing that stands out, is that n_{ins} does not result in a significant improvement of the solution while n_{ratio} and n_{swap} do. To conclude, n_{ins} will not be used further in this research, and

Neighbourhood	n_{ins}			n_{ratio}			n_{swap}		
Tabu list length	30	50	70	10	20	30	30	50	70
Improvement of solution (%)	2.76	3.05	3.36	6.63	7.66	8.02	9.40	10.55	8.77
Running time (m)	11.36	18.06	17.74	1.81	1.68	1.86	38.93	60.28	57.76
Running time per iteration (m)	0.38	0.60	0.59	0.06	0.06	0.06	1.30	2.01	1.93
Iteration with best objective value	29.17	28.67	29.00	18.83	24.83	26.00	23.33	25.33	20.67

Table 6: Results TS for the neighbourhoods with different tabu list settings

the tabu list length of n_{ratio} and n_{swap} will both be set to 30. This choice is made by looking at both the improvement and the running time per iteration. For n_{ratio}, the length of 30 results in the best outcome and the running time is equal for each setting. For n_{swap} the length of 50 results in the best outcome, but the running time per iteration is much higher than for length 30, which brings us to the choice of a tabu list length of 30, which provides a comparable outcome.

In general, the more local search iterations performed, the more the solution improves. Table 6 shows that for both n_{swap} and n_{ratio} the number of iterations in which improvement occurs lays around 26 and 23 iterations respectively. The number of iterations for these neighbourhoods is based on these numbers.

The reason to choose two neighbourhoods to continue the research on instead of one is that it is important to explore the solution space thoroughly which might be achieved by using multiple neighbourhoods. The next section provides an analysis of combining the neighbourhoods.

6.2.1 Combining neighbourhoods

To search a larger area of the solution space, the neighbourhoods investigated in this research will be combined. This can be done by performing multiple different neighbourhoods consecutively on the solution. A lot of options for the number of iterations and the order of neighbourhoods exist, thus a selection of possibilities to test is made. The goal of this experiment is to get an insight on which combination of the neighbourhoods n_{swap} and n_{ratio} improves the solutions the most in a reasonable amount of time.

A possible combination sequence is denoted as $[n_1|i_{n_1}, n_2|i_{n_2}, n_3|i_{n_3}, n_4|i_{n_4}]$, where n_j denotes the j^{th} neighbourhood in the sequence, and i_{n_j} the amount of iterations executed. The sequence options that are tested are denoted in Table 7, which also contains the improvement and running times of the solutions. The sequences are tested on all the instances of INS6, and the average of the results is reported.

When analysing the results in Table 7, options 2, 5, 6, 8 and 10 result in the most improvement of the solution. It is important to take the running time into account since the final algorithm might include

Option	n_1	i_{n_1}	n_2	i_{n_2}	n_3	i_{n_3}	n_4	i_{n_4}	**Improvement of solution (%)**	**Running time (m)**
1	n_{ratio}	15	n_{swap}	3	-	-	-	-	6.45	4.84
2	n_{ratio}	30	n_{swap}	6	-	-	-	-	10.38	16.61
3	n_{swap}	3	n_{ratio}	15	-	-	-	-	6.02	4.41
4	n_{swap}	6	n_{ratio}	30	-	-	-	-	8.52	10.18
5	n_{swap}	3	n_{ratio}	10	n_{swap}	3	n_{ratio}	10	8.31	8.78
6	n_{swap}	10	n_{ratio}	3	n_{swap}	10	n_{ratio}	3	9.63	32.39
7	n_{swap}	3	n_{ratio}	3	n_{swap}	3	n_{ratio}	3	6.61	11.66
8	n_{swap}	6	n_{ratio}	6	n_{swap}	6	n_{ratio}	6	8.33	21.55
9	n_{ratio}	3	n_{swap}	10	n_{ratio}	3	n_{swap}	10	7.57	22.93
10	n_{ratio}	10	n_{swap}	3	n_{ratio}	10	n_{swap}	3	9.58	20.81
11	n_{ratio}	3	n_{swap}	3	n_{ratio}	3	n_{swap}	3	5.35	9.28
12	n_{ratio}	6	n_{swap}	6	n_{ratio}	6	n_{swap}	6	7.79	13.08

Table 7: Results TS with different neighbourhood sequences

many iterations.

When evaluating the results in Table 6, we find that the most promising combinations have high running times and comparable improvement to the results found in Table 6, where TS is run for 30 iterations. To conclude, it is not profitable to combine the neighbourhoods, the running time increases and the results are comparable to using TS with n_{ratio} for 30 iterations.

6.3 ACO and TS combined

The goal of the experiments performed is to decide on the settings of the final algorithm which combines ACO and TS. Now that the ACO and TS settings are decided on, the combination of the two algorithms is studied. In Section 4.4, we elaborated on two options for the combination; TS iteration in each ACO iteration, or updating the visibilities using a tabu list. The second option was already tested and found to not be efficient, thus, we continue with the first option and perform the TS in each ACO iteration. An overview of the settings for the algorithm can be found in Table 8. The algorithm is executed using these settings on the instances of INS6, the average results can be found in Table 9.

6.4 Boosting initial solution population

Section 4.3 introduces three boosting methods for the initial solution population. This section reports the results of the experiments on the quality of the boosting methods and their different possible settings. These experiments all have the same structure; the boosting method is performed after which the ACO and TS combination algorithm is run on the instances of INS6 and the average of all the results is reported.

ACO	
Updating visibilities option	Worst route
Number of ants (n_{ants})	5
Evaporation rate (ρ)	0.3
Number of routes to fixate	1
Number of iterations	30

TS	
Neighbourhood	n_{ratio}
Number of iterations	26
Tabu list length	30

Table 8: Overview of the settings decided on by the experiments

ACO and TS combined	
Improvement of solution (%)	19.95
Running time (m)	288.10
Running time per ACO iteration (m)	11.14

Table 9: ACO and TS combined

Multi-Colony boost

Based on the results from earlier experiments, the neighbourhoods used in this method are n_{swap} and n_{ratio}, thus two colonies are created in the solution population. The neighbourhood n_{swap} is performed 23 times and the neighbourhood n_{ratio} is performed 26 times, these numbers are based on the findings of experiments on the neighbourhoods reported in Table 6. The Multi-Colony algorithm is run for three iterations, where after the ACO combined with TS is performed. Table 10 contains the results for the algorithm including the boosting process.

Tabu Search boost

The results from earlier experiments are used to decide on the settings of the experiments (8). The proportion of the ants that the boosting is performed on (p_{TS}) and the range from which the number of iterations is chosen from needs to be decided on. The range is 1 to 26 since these are the number of iterations in which improvement takes place according to Table 6. The proportion p_{TS} is tested for the values 40% and 60%. The results of these experiments are reported in Table 10.

Simulated Annealing election

From earlier experiments, the amount of iterations that are performed on the solutions is set to 26 (Table 6). The probability of the worst solution to be chosen (p_{worst}) is tested for 20% and 40%. The results for the different settings of this boosting method are reported in Table 10.

Boost method	**Multi-Colony**	**TS boost**		**SA election**	
Parameter	-	p_{TS}		p_{worst}	
Values	-	40%	60%	20%	40%
Improvement of solution ($\%$)	18.30	18.62	19.27	20.06	18.83
Total running time (m)	358.14	339.74	283.41	326.44	296.66
Boosting running time (m)	55.74	0.32	1.65	4.18	5.35

Table 10: Results boosting initial solution population methods

comparing the results in Table 9 and Table 10, the improvement of the solution including boosting methods the does not seem to be significantly higher than without. From Table 9 we find that the algorithm improves the solution by 19.95% without boosting, and from Table 10 we find that this is at most 20.06%. The running time increases by 4.18 minutes, and due to the randomness in the algorithm that might have caused the 0.11% difference in the outcome, we decide to not include the boosting method.

Now that we have decided on all the settings, extensions and boosting method, the algorithm is finalised as illustrated in Figure 5.

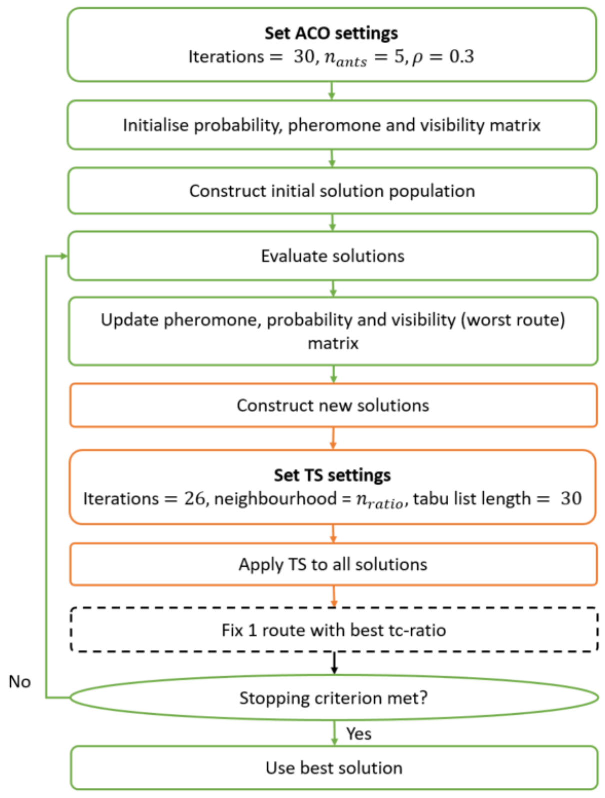

Set ACO settings
Iterations = 30, n_{ants} = 5, ρ = 0.3
Initialise probability, pheromone and visibility matrix
Construct initial solution population
Evaluate solutions
Update pheromone, probability and visibility (worst route) matrix
Construct new solutions
Set TS settings
Iterations = 26, neighbourhood = n_{ratio}, tabu list length = 30
Apply TS to all solutions
Fix 1 route with best tc-ratio
No
Stopping criterion met?
Yes
Use best solution

Figure 5: Final ACO and TS combined algorithm

7 Testing on different data sets

This section provides an evaluation of the algorithm for the different types of data sets INS_U, INS_10d, INS_30d, INS_eq, INS_low and INS_high. The algorithm is tested on each data set once, the outcome is reported in Table 11.

Data set	**INS_U**	**INS_10d**	**INS_30d**	**INS_eq**	**INS_low**	**INS_high**
Improvement of solution (%)	5.05	13.53	18.56	2.03	23.76	8.07
Running time (m)	330.23	350.48	263.16	380.26	183.73	733.28

Table 11: Results for different types of data sets

The values reported in Table 7 show that the algorithm does not improve the initial solutions for the data sets 'Urban area', 'Equally distributed visiting frequencies' and 'High visiting frequencies'. For the 'Urban area', this is because customers lay within a smaller area, which causes less room for improvement of the objective value, total travel time. The 'Equally distributed' and 'High visiting frequencies' data sets contain more customers with $f_{v_i} \in \{8, 12, 16, 20\}$, which causes fewer options for visiting schedules per customer and thus less room for optimisation. Also, the visiting schedules of these customers contain many visiting days, which forces more customers to be placed on a specific day, and thus less flexibility. For the '10 day planning horizon' there exist fewer possible visiting schedules and there are fewer days to plan routes on. The '30-day planning horizon' data set has an improvement comparable to that of the original instances since the customers have the same amount of possible visiting schedules. The running time is lower since the number of customers per day is less, thus routing takes less time. For the 'Low visiting frequencies' we find the most improvement, this makes sense since customers with a low visiting frequency have possible visiting schedules containing few days, so fewer customers are forced to specific visiting days.

To conclude this section, the algorithm is not suitable for rural areas and customers with equally distributed or high visiting frequencies, there is not enough room for improvement and the visiting schedules belonging to these data sets do not provide enough flexibility. The outcome of the algorithm is comparable to that of the original instances, and thus it is a suitable algorithm for those data sets.

8 Conclusion

The problem addressed in this research is the Periodic Vehicle Routing Problem, an efficient routing problem over a certain planning horizon. An algorithm solving this problem consists of two parts: assigning customers to visiting days and solving the Vehicle Routing Problem (VRP) on these visiting days. Customers indicate how often they want to be visited in the planning horizon, the amount of waste is to be collected and in what time range this is possible. The biggest challenge of solving the PVRP is the assignment of customers to visiting days, past literature on the VRP is used for the second phase and is not investigated in this research. In this thesis, the PVRP is converted to the Machine Scheduling Problem (MSP) such that solving methods for the MSP can be used as an inspiration for the algorithm to solve the PVRP. This section provides a summary on how the idea of the algorithm arose and on the experimental set up. Then, the results, limitations and lastly some ideas for further research are discussed.

This thesis presents a literature study to give an overview of potential solving methods for the MSP, and thus the PVRP. Both the PVRP and MSP have been studied extensively in literature, often a semi-random constructive heuristic is combined with a local search heuristic. This literature study also shows that the Ant Colony Optimisation (ACO) and Tabu Search (TS) are popular methods for solving the MSP, which makes it interesting to investigate combining them.

Two possible ways for combining the algorithms are investigated, one option includes TS in a specific element of the ACO, the other performs TS after each iteration of ACO. Inspired by literature, some extensions for the algorithm are implemented to improve the combination of ACO and TS. Both ACO and TS have many parameters that have to be tuned, thus together with the extensions a lot of options for the settings arise. At first, the settings and extensions for the ACO are tested and decided on, then those of TS are determined. With these settings, the final algorithm is formed and the initial solution population boosting methods are tested.

All these settings are decided on by conducting experiments with instances based on the original data set provided by GreenRoutes. To be able to compare the outcome of the algorithm for the different instances, the improvement of the solution is quantified. This is done by expressing the improvement of the algorithm as a percentage which is based on the initial objective value and the objective value of the best solution found by the algorithm.

As mentioned, the main goal of the thesis is to find out whether the algorithm is suitable to solve the PVRP. The results from the experiments show that for the instances based on the data set provided by GreenRoutes, the combined ACO and TS algorithm can improve the solutions by 19.95% in 288.10 minutes (Table 9). This indicates that the algorithm is suitable, but when looking at the ACO and TS separately, one might question whether the combination performs better than the two algorithms separately. Individually, the improvement by the ACO using 30 ants is 19.76% in 115.63 minutes (Table

2, without ‘fixating routes’ extension), this improvement might increase when the algorithm is ran for more iterations. When looking at the TS individually, we see that when 100 iterations of TS are performed on a solution, the improvement is 12.85%, while the running time is only 7.47 minutes. We can conclude that the combined algorithm is suitable for the PVRP, but does not significantly outperform the separate algorithms ACO and TS. It is interesting to investigate the separate algorithms, since they cost less computing time and have a lot of potential to increase improvement further when researched.

What stood out when testing the ACO, is that increasing the amount of ants does not increase the improvement of the solution significantly. This was not expected since using more ants results in more high quality $c - s$ combinations in the solution population which might appear in the next solution population. Using less ants in the algorithm results in much less running time, so despite this was unexpected, the outcome is positive. The aspect that had the most impact on the improvement of the solution was the method to update the visibilities, for two visibility updating methods the increase of improvement was 0.69% or 2.11%, while for one this was 19.76%. This indicates that this stage of the algorithm is very important when solving the PVRP using ACO.

What was remarkable for the TS was that we found that the neighbourhood with the least running time provided very good results (n_{ratio}). This neighbourhood took the quality routes into account, which seems an efficient way to improve the solution. The neighbourhoods n_{ins} and n_{swap} required long running time since it took a long time to find the best possible solution in those neighbourhoods that was not tabu, and did not outperform n_{ratio}. Another aspect that stood out, is that the different length sizes of the tabu list did not result in significant difference for the outcome.

Throughout the research, it became obvious that the running time for the combination of ACO and TS is too long, even when the algorithm was tested on instances smaller than the original. Running the algorithm on smaller instances was fine for the goal of the thesis, to find out whether the algorithm is suitable for the PVRP, but it indicates that the algorithm is not compatible with instances as big as the original data set.

Since the algorithm contains random elements it is hard to compare the results of the same instances for different settings. Due to the running time, it was not possible to run the algorithm more often, which might have resulted in more reliable results. There exists no benchmark instance for the PVRP with the specific planning horizon and visiting frequencies from this data set, thus it was not possible to compare the results from our algorithm to those discussed in literature, which makes it harder to decide on the quality of the algorithm overall. From Section 7 we find that the algorithm does not perform as well for data sets with a different distribution of visiting frequencies among the customers, which indicates the algorithm with the current settings might not be suitable for different data sets.

Next to the many different possible settings in the algorithm, a lot of parameters in the ACO were not tested but set to values retrieved from literature. For example α and β in the pheromone levels and Q, R and q in the visibility updating stage. Tuning these parameters might have resulted in more improvement of the results.

For further research on the ACO, all the parts of the algorithm can be investigated separately. For example, the different visibility updating methods discussed in this thesis caused very different outcomes of the algorithm. These visibility updating methods can be based on those for solving the MSP, but one can also investigate other visibilities which favour or disfavour combinations appearing in certain routes.

For the TS, many more options for neighbourhoods can be investigated. Also, other types of tabu lists are interesting. The type investigated in this research is to add customers or routes to a list, one can also decide to add parts of routes or $c - s$ combinations to a tabu list, resulting in more flexibility when constructing solutions. Since literature shows that local search heuristics are popular to improve solutions for both the MSP and the PVRP, investigating different types of local search is also recommended.

References

Arnaout, J. P., Musa, R. & Rabadi, G. (2014), 'A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines - part ii: Enhancements and experimentations', *Journal of Intelligent Manufacturing* **25**, 43–53.

Arnaout, J. P., Rabadi, G. & Musa, R. (2010), 'A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times', *Journal of Intelligent Manufacturing* **21**, 693–701.

Arnaout, J., Rabadi, G. & Musa, R. (2008), 'Ant colony optimization algorithm to parallel machine scheduling problem with setups', *International Conference on Automation Science and Engineering* p. 1030.

Bell, J. E. & McMullen, P. R. (2004), 'Ant colony optimization techniques for the vehicle routing problem', *Advanced Engineering Informatics* **18**, 41–48.

Cordeau, J.-F., Laporte, G. & Mercier, A. (2001), 'A unified tabu search heuristic for vehicle routing problems with time windows', *Journal of the Operational Research Society* **52**, 928–936.

Dantzig, G. B. & Ramser, J. H. (1959), 'The truck dispatching problem', *Management Science* **6**, 80–91.

Flood, M. M. (1956), 'The travelling-salesman problem', *Operations Research* **4**, 61–75.

Gmira, M., Gendreau, M., Lodi, A. & Potvin, J. Y. (2021), 'Tabu search for the time-dependent vehicle routing problem with time windows on a road network', *European Journal of Operational Research* **288**, 129–140.

Google (n.d.*a*), pp. [Map of Rotterdam, The Hague and surroundings.]. Retrieved October 25, 2022, from `https://www.google.com/maps/place/Delfgauw/`.

Google (n.d.*b*), pp. [Map of Delfgauw]. Retrieved October 25, 2022, from `https://www.google.com/maps/@51.9897286,4.1984482,10z`.

Helal, M., Rabadi, G. & Al-Salem, A. (2006), 'Unrelated parallel machines scheduling problem with setup unrelated parallel machines scheduling problem with setup times times', *International Journal of Operations Research* **3(3)**, 182–192.

Lenstra, J. K. & Kan, A. H. G. R. (1981), 'Complexity of vehicle routing and scheduling problems', *NETWORKS* **11**, 221–227.

Li, X. Y., Tian, P. & Leung, S. C. (2009), ‘An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems’, *Journal of the Operational Research Society* **60**, 1012–1025.

Mancini, S. (2017), ‘A combined multistart random constructive heuristic and set partitioning based formulation for the vehicle routing problem with time dependent travel times’, *Computers and Operations Research* **88**, 290–296.

Mat, N. A., Benjamin, A. M., Abdul-Rahman, S. & Wibowo, A. (2017), ‘Nearest greedy for solving the waste collection vehicle routing problem: A case study’.

Michallet, J., Prins, C., Amodeo, L., Yalaoui, F. & Vitry, G. (2014), ‘Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services’, *Computers and Operations Research* **41**, 196–207.

Pirkwieser, S. & Raidl, G. R. (2008), ‘A vns for the pvrptw a variable neighborhood search for the periodic vehicle routing problem with time windows’.

Solomon, M. M. (1987), ‘Algorithms for the vehicle routing and scheduling problems with time window constraints.’, *Operations Research* **35**, 254–265.

Tan, K. C., Lee, L. H., Zhu, Q. L. & Ou, K. (2000), ‘Heuristic methods for vehicle routing problem with time windows’, *Artificial Intelligence in Engineering* **15**, 281–295.

Thangiah, S. R. (1995), ‘Vehicle routing with time windows using genetic algorithms’.

Thangiah, S. R. & Salhi, S. (2001), ‘Genetic clustering: An adaptive heuristic for the multidepot vehicle routing problem’, *Applied Artificial Intelligence* **15**, 361–383.

Udomsakdigool, A. & Kachitvichyanukul, V. (2008), ‘Multiple colony ant algorithm for job-shop scheduling problem’, *International Journal of Production Research* **46**, 4155–4175.

Wang, Y., Wang, L., Chen, G., Cai, Z., Zhou, Y. & Xing, L. (2020), ‘An improved ant colony optimization algorithm to the periodic vehicle routing problem with time window and service choice’, *Swarm and Evolutionary Computation* **55**.

Ying, K. C., Lee, Z. J. & Lin, S. W. (2012), ‘Makespan minimization for scheduling unrelated parallel machines with setup times’, *Journal of Intelligent Manufacturing* **23**, 1795–1803.

Yu, B. & Yang, Z. Z. (2011), ‘An ant colony optimization model: The period vehicle routing problem with time windows’, *Transportation Research Part E: Logistics and Transportation Review* **47**, 166–181.

A Visiting schedules per visiting frequency

Visiting schedules per visiting frequency f_i for different planning horizon. The 20 day planning horizon is standard, the 10 and 30 day planning horizon is used to test the performance of the algorithm for different planning horizons.

f_i	S_{f_i} **Planning horizon of 10 days**
1	{1} {2} {3} {4} {5} {6} {7} {8} {9} {10}
2	{1, 6} {2, 7} {3, 8} {4, 9} {5, 10}
4	{1, 4, 6, 9} {1, 5, 6, 10} {2, 5, 7, 10}
6	{1, 2, 5, 6, 7, 10} {1, 3, 4, 6, 8, 9} {1, 3, 5, 6, 8, 10} {1, 4, 5, 6, 9, 10}
8	{1, 2, 3, 5, 6, 7, 8, 10} {1, 2, 4, 5, 6, 7, 9, 10} {1, 3, 4, 5, 6, 8, 9, 10}
10	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

f_i	S_{f_i} **Planning horizon of 20 days**
1	{1} {2} {3} {4} {5} {6} {7} {8} {9} {10}
	{11} {12} {13} {14} {15} {16} {17} {18} {19} {20}
2	{1, 11} {2, 12} {3, 13} {4, 14} {5, 15}
	{6, 16} {7, 17} {8, 18} {9, 19} {10, 20}
4	{1, 6, 11, 16} {2, 7, 12, 17} {3, 8, 14, 18} {4, 9, 14, 19} {5, 10, 15, 20}
8	{1, 4, 6, 9, 11, 14, 16, 19} {1, 5, 6, 10, 11, 15, 16, 20} {2, 5, 7, 10, 12, 15, 17, 20}
12	{1, 2, 5, 6, 7, 10, 11, 12, 15, 16, 17, 20} {1, 3, 4, 6, 8, 9, 11, 13, 14, 16, 18, 19}
	{1, 3, 5, 6, 8, 10, 11, 13, 15, 16, 18, 20} {1, 4, 5, 6, 9, 10, 11, 14, 15, 16, 19, 20}
16	{1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 15, 16, 17, 18, 20}
	{1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 19, 20}
	{1, 3, 4, 5, 6, 8, 9, 10, 11, 13, 14, 15, 16, 18, 19, 20}
20	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

f_i	S_{f_i} **Planning horizon of 30 days**
1	{1} {2} {3} {4} {5} {6} {7} {8} {9} {10}
	{11} {12} {13} {14} {15} {16} {17} {18} {19} {20}
	{21} {22} {23} {24} {25} {26} {27} {28} {29} {30}
3	{1, 11, 21} {2, 12, 22} {3, 13, 23} {4, 14, 24} {5, 15, 25}
	{6, 16, 26} {7, 17, 27} {8, 18, 28} {9, 19, 29} {10, 20, 30}
6	{1, 6, 11, 16, 21, 26} {2, 7, 12, 17, 22, 27} {3, 8, 14, 18, 23, 28}
	{4, 9, 14, 19, 24, 29} {5, 10, 15, 20, 25, 30}
12	{1, 4, 6, 9, 11, 14, 16, 19, 21, 24, 26, 29} {1, 5, 6, 10, 11, 15, 16, 20, 21, 25, 26, 30}
	{2, 5, 7, 10, 12, 15, 17, 20, 22, 25, 27, 30}
18	{1, 2, 5, 6, 7, 10, 11, 12, 15, 16, 17, 20, 21, 22, 25, 26, 27, 30}
	{1, 3, 4, 6, 8, 9, 11, 13, 14, 16, 18, 19, 21, 23, 24, 26, 28, 29}
	{1, 3, 5, 6, 8, 10, 11, 13, 15, 16, 18, 20, 21, 23, 25, 26, 28, 30}
	{1, 4, 5, 6, 9, 10, 11, 14, 15, 16, 19, 20, 21, 24, 25, 26, 29, 30}
24	{1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 15, 16, 17, 18, 20, 21, 22, 23, 25, 26, 27, 28, 30}
	{1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 19, 20, 21, 22, 24, 25, 26, 27, 29, 30}
	{1, 3, 4, 5, 6, 8, 9, 10, 11, 13, 14, 15, 16, 18, 19, 20, 21, 23, 24, 25, 26, 28, 29, 30}
30	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30}

Table 12: Visiting frequencies and the respective visiting schedules when planning horizon is 10 days

B Map original data set

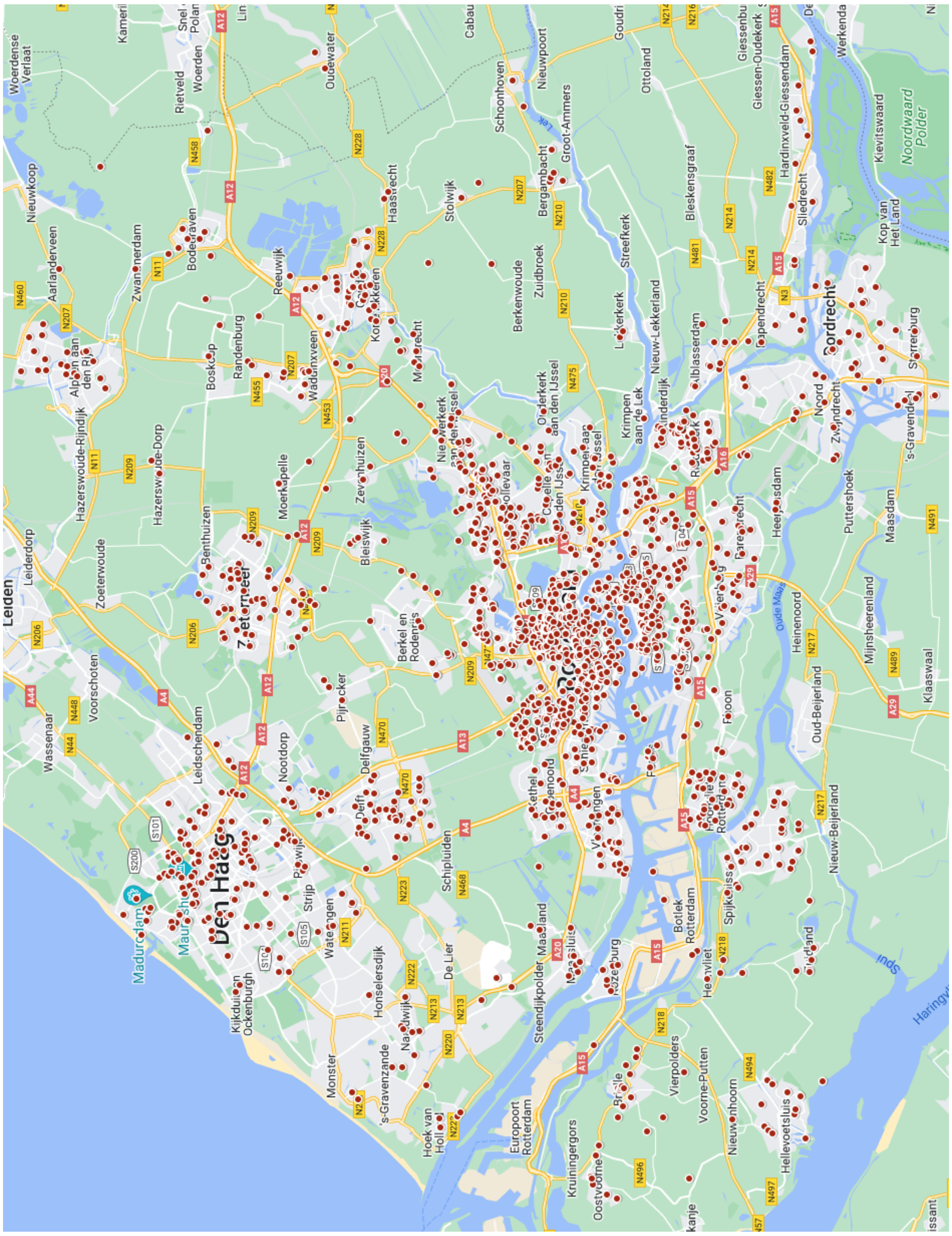


Figure 6: Locations of all customer included in the original data set provided by GreenRoutes (Google n.d.*a*)

C Map urban area

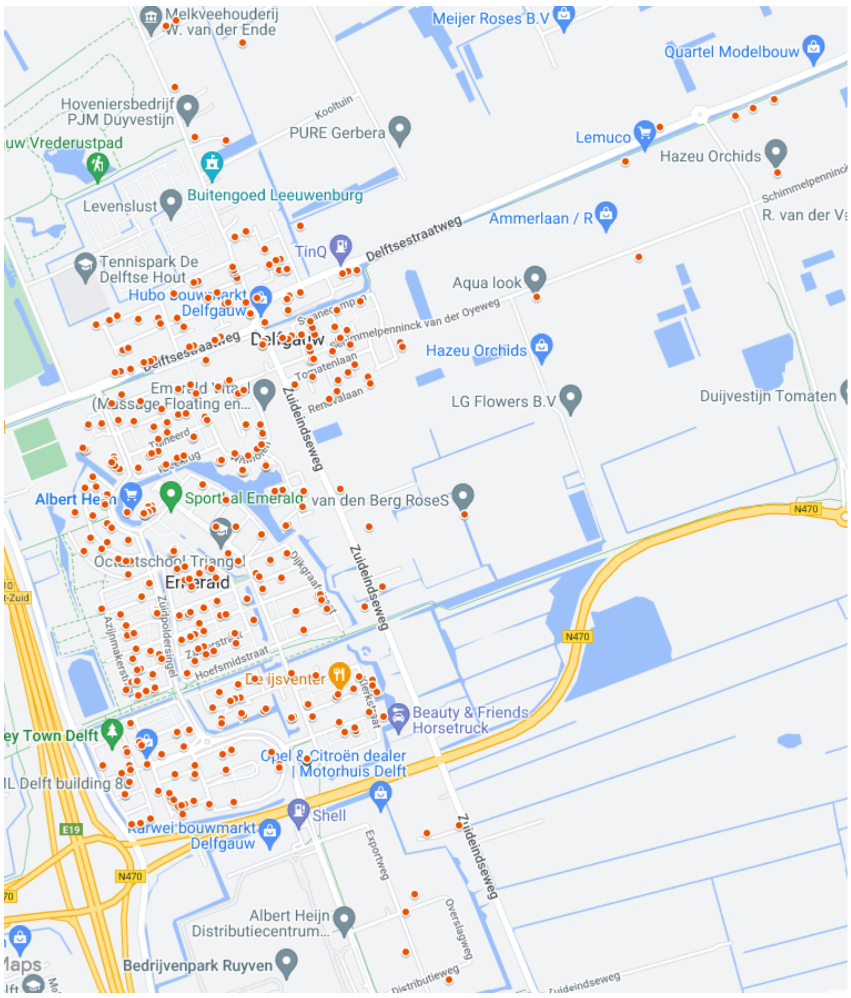


Figure 7: Locations of all customer included in the urban area data set elaborated on in Section 5.2 (Google n.d.*b*)

D Algorithm structures ACO and TS

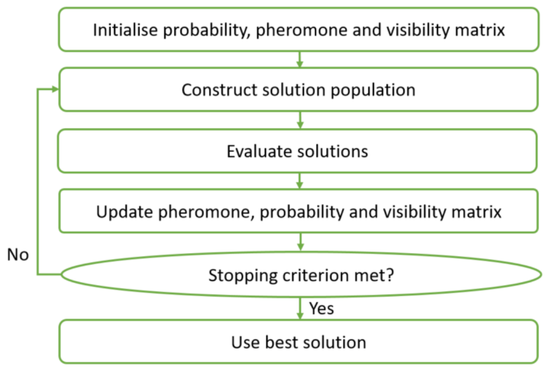

Initialise probability, pheromone and visibility matrix
Construct solution population
Evaluate solutions
Update pheromone, probability and visibility matrix
No
Stopping criterion met?
Yes
Use best solution

Figure 8: ACO flowchart

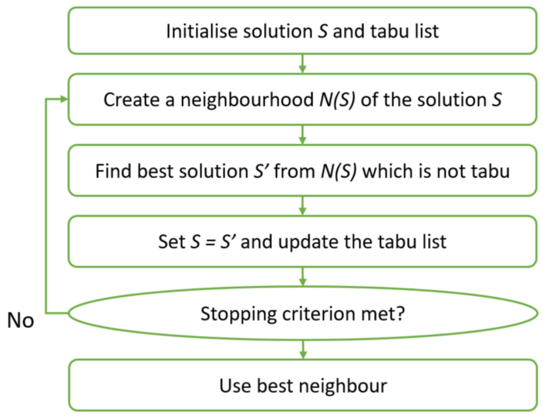

Initialise solution S and tabu list
Create a neighbourhood N(S) of the solution S
Find best solution S' from N(S) which is not tabu
Set S = S' and update the tabu list
No
Stopping criterion met?
Use best neighbour

Figure 9: TS flowchart