Universiteit Leiden
ICT in Business

Solving Dynamic Vehicle

Routing Problems in Practice

Name: J.P. van Osta

Date: 16/08/2014

1st supervisor: Michael Emmerich

2nd supervisor: Thomas Bäck

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

The Dynamic Vehicle Routing Problem (DVRP) is a routing problem with multiple vehicles which changes routes during the execution of the problem. With our current technologies such as GPS and mobile internet applications these problems can be put to practice more effective than before. However, there is a gap between state-of-the-art theory and practice. Van Veen et al. [24] present an Ant Colony Optimization algorithm designed to solve a DVRP with time windows. This research is based around implementing this algorithm to test it and use it in a real-world application. As test case a security company was used. This security company has a number of a-priori known jobs and a (smaller) number of unpredictable incidents. Both need to be visited within a certain time frame. The requirements were formulated after an interview with a data analyst. The algorithm with additional requirements was implemented on an agent communicating with a logistics (delivery) system. Two trials were conducted testing the algorithm thoroughly. Involved were a number of drivers with mobile applications. The end results proved positive, reducing the total driving time with 8.6%. The implementation was not completely perfect, still some orders were not visited on time, but some possible future extensions to the algorithm are given to improve its performance. The general learned lessons were: Implementing practice means testing in practice, working iteratively leads to success, communication is key, people are important.

Contents

1 Introduction

A Vehicle Routing Problem (VRP) is a problem where one or multiple vehicles have to be assigned to multiple customers or addresses. The *Dynamic* Vehicle Routing Problem is one where these routes may change during the execution of these routes, due to changing information. Pillac et al. [16] discuss the rise of certain technologies in recent years (such as Intelligent Transport Systems) and that these technologies have made it possible to implement certain Dynamic Vehicle Routing applications. The main applications mentioned are: Transport of goods (changing traffic or changing customers), Transport of persons (taxis etc.) and services. This last application will be the main focus of this research.

Fu [9] argues that there is a gap between the theoretical research done in simulation optimization and the software which is actually used in practice. A lot of optimization algorithms which are being developed and researched perform very well on benchmark problems but lack the robustness to be used on stochastic real-world problems. This results in applications having to use relatively general or relatively simple strategies to accomplish their goals.

Ant Colony Optimization (ACO) is a metaheuristic and can therefore be applied to a wide variety of problems, including the VRP. It is relatively robust and usually produces good results in a relatively short amount of time. For instance, Bell et al. [1] present an ACO algorithm which finds solutions within 1% of the optimal solution for a number of VRPs. Van Veen et al. [24] present an Ant Colony Optimization algorithm specifically designed for Dynamic Vehicle Routing Problems. This Multiple Ant Colony System performs very well on the Solomon benchmark problems [21]. However, it has never been put to practice in a real-world situation. A stochastic real-world test case might need a more robust and specific implementation than a controlled simulated environment does.

In this research the algorithm of [24] was implemented so it could be tested on a real-world problem. This dynamic VRP was based on a security company with hundreds of customers each day, some planned beforehand and some scheduled during the day. Two trials were conducted in the Rotterdam area to test this algorithm. By doing this we will try to answer the following research questions:

1. How can the algorithm of [24] be successfully put to practice?
2. What should be considered when implementing a (dynamic routing) algorithm for a real-world environment?

The structure of the remainder of this document will be as follows. First of all, a problem description is given, discussing the VRP and the business case of the security company. Then, Ant Colony Optimization will be discussed, explaining the basic algorithm and some extensions of it. The algorithm of [24] will also be explained. After this, the research methods will be explained, including the implementation process and the details of the practical trials. Then the results will be presented, followed by the discussion and conclusion. This document will end with some suggestions on further work. For any variables or abbreviations, a nomenclature is included.

2 Problem description

The main intention of this research is to implement a Dynamic Vehicle Routing Problem into practice using Dynamic Ant Colony Optimization algorithms. This section will focus on getting a better grip on this problem domain by first explaining some Vehicle Routing Problem types. For practical testing a real-world case is needed. This business case will be described in detail. The relevance of this work will also be discussed.

2.1 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is one of the most researched problem types within logistics. According to [23], hundreds of algorithms and other methods were introduced to solve the VRP since its introduction in 1959. [5] The problem is widely studied because of its relevance to (cost) efficiency and its relative complexity, especially when the problem set grows.

2.1.1 Basic Vehicle Routing Problem

Before the relevant VRP variant for this research can be discussed it is important that we establish the basics of the most elementary type of the VRP. The three most important characteristics of the VRP are:

1. The depot, the starting point of all vehicles.
2. r, the number of vehicles. $r > 0$.
3. n, the number of addresses or customers. $n \geq r$.

If $r = 1$ we are dealing with a Traveling Salesman Problem, (TSP) which is technically the most basic version of a VRP. For the sake of explaining the VRP we will assume that most VRPs are complex enough so that $r > 1$.

The main goal of solving the VRP is to assign routes to the vehicles so that all customers can be visited as efficiently as possible. Efficient can mean different things. According to [23] typical objectives are:

- Minimization of the global transportation cost, including total distance and labor time.
- Minimization of the number of vehicles.
- Balancing the routes, by distributing the work equally among vehicles.

- Minimization of the penalties associated with partial service of customers, such as late fines.

It should be noted that some goals are closely related. For instance: when minimizing the total number of vehicles, this could also lower the total amount of transportation costs. And, when optimizing route lengths, this could automatically prevent tardiness and thus late fines.

Toth et al. [23] describe the notation of the VRP as follows: A complete graph $G = (V, A)$ where $V = \{0, ..., n\}$ corresponds to the depot (0) and the customers $(1, ..., n)$. A is the arc set. There is a nonnegative cost c_{ij} for every line $(i, j) \in A$. As mentioned before, the usual goal is to minimize these costs while visiting each customer. In the very basic form of the VRP all customers are available at all times.

A very common variant of the VRP is the capacitated VRP, or CVRP, where each vehicle has an equal maximum capacity for packages, or customers it can serve. However, for the purposes of this research, that is scheduling routes for a security company without delivery of goods, the CVRP will not need to be discussed any further.

2.1.2 VRP with Time Windows

In the previous subsection the basic VRP was discussed, where each customer was always available. However, this is not always the case. Sometimes certain customers can only open the door at specific hours, or contractual time-frame obligations have to be adhered. In this case we are speaking of a VRP with Time Windows, a VRPTW.

Each customer i is given a time window $[a_i, b_i]$. Within this time window the delivery should start. To preemptively calculate the customer sequence there should be a travel distance (or time) d_{ij} between each pair of customers i and j. d_{ij} could possibly replace c_{ij}. Each customer i will also have a service time s_i, where $s_i \geq 0$. This is the time it takes to visit the customer. After this time, the vehicle can go on its way to visit the next address.

2.1.3 Dynamic VRP

The final extension to the VRP which will be introduced in this problem description is the Dynamic VRP, or DVRP. The main characteristic of this extension is that not all customers are known beforehand. During the execution of the initial solution of the VRP, more customers will be added to the problem set.

n will grow and thus G will grow. A portion D of the customers will not be available initially and will be introduced one by one at a given time t_i. The remaining fraction of customers will be known a-priori. $((1 - D) *$ number of customers) D is the *dynamicity* of the problem and can be expressed as either a fraction or a percentage. Introducing new customers will make the problem more complex and it also requires a certain amount of slack-time within the initial calculations at $t = 0$, allowing for growth. Sometimes, an estimation of D is known a-priori, which can be used to determine the amount of (initial) slack. However, in practice the exact value of D is almost never known a-priori.

Psaraftis [17] informs us that the possibilities for applications of the DVRP have grown with the advancement of location-based technologies such as GPS and GIS (Geographical Information Systems). These technologies can, in combination with today's communication possibilities, be utilized to effectively refer vehicles to new, unexpected customers. Therefore, researching the DVRP has become very relevant, as well as the DVRP variant with time window constraints: the DVRPTW.

2.2 Business case

2.2.1 Case study

To put the theory into practice, we need a real-life problem, a business case. A business case was derived from a business where a DVRP with Time Windows was present. Consider a security company. Every day this security company has between 300 and 400 planned jobs in the Rotterdam area. These planned jobs include surveillance, security checks and the opening or closing of buildings, among others. There are clear contracts about the time windows and tasks which are included in such a job. Also, the average service time for each job is known. The deviation, along with a typical minimum and maximum service time are also well-known. This is based on historic data. Each day, there is an average of about 45 incidents (or alarms) in this same region. However, this amount can vary from 30 to 110 incidents. These incidents can for instance be fire alarms, burglary alarms or technical problems. They appear during the day and cannot be predicted. Some predictions can be made, i.e. most alarms occur in the evening and on industrial terrains, but we do not know their exact times and other properties beforehand. Therefore this business case is perfect to implement a DVRPTW. This DVRPTW has an average dynamicity of 11.6%.

To translate this case into a VRP we need an objective function to be optimized. Multiple objectives were considered. Cost minimization could be

performed on (for instance) fuel cost, total distance, labor time or the number of fines. Fines will originate from *contract violations* such as visits outside of time window constraints. The ideal minimization would consider all of these factors and minimize the total costs. However, this would make the problem a lot more complex. Of course, there is a big correlation between total distance, fuel, and time. So by minimizing one of these factors we can expect to obtain approximately optimal values for the other factors. Also, a time window algorithm will always try to work within the given time windows, automatically minimizing fines. Most costs originate from *total labor time*, given the fact that the contract violation fines do not exceed the savings on labor cost. Therefore the main goal should be minimizing the labor time. An important constraint is that a work shift usually should take 3 to 8 hours. Between 8 to 10 hours overtime is paid. Less than 3 hours or more than 10 hours is not allowed. Using as less vehicles as possible is also desirable.

2.2.2 Current solution

At the moment there is almost no dynamicity implemented in the baseline algorithm used in the business case. All jobs which are known a-priori, the static jobs, are scheduled by a state-of-the-art static VRPTW algorithm. The exact algorithm is unknown to us, as it is confidential. Also, a number of vehicles is always on stand-by. Their job is solely to react to any incoming incidents. Incidents are assigned by a (human) coordinator. In most cases an incident will go to the closest stand-by vehicle. In very rare cases, an incident will be picked up by a static job vehicle. The coordinator might need to do some manual rescheduling in this case.

This approach has some drawbacks:

- The response to incidents might be too late if all incident vehicles are busy at the same time.
- It takes time for the coordinator to plan all the incidents. Especially when multiple incidents come in at once and routes need to be rescheduled.
- On a quiet day (a day with less than average incidents), the incident vehicles will be idle most of the time. This results in unnecessary labor time and bored employees.

Possible upsides of such an approach are:

- Static job vehicle drivers know exactly what they have to do all day. This can make them more efficient and/or confident.

- Incident vehicle drivers can specialize themselves in handling incidents. Training costs could be cheaper as apposed to a dynamic solution where all employees should be able to respond to any type of customer.

2.2.3 Case concepts

Each customer (or job) i has the following properties:

- A location. This is an address. The travel time, cost or distance d_{ij} between two jobs i and j can be calculated by a navigation (web)service, such as Google Maps.
- A service time s_i. The time it takes to complete the job. The service time is not always known a-priori. Sometimes a job takes unexpectedly long or short. I.e. when a burglary alarm turns out to be a false alarm.
- A time window $[a_i, b_i]$. The security company is contractually obliged to visit within this time frame. Most time windows have an interval of multiple hours, some less than an hour. An incident time window is either 30 or 45 minutes.
- A priority p, ranging from 1 to 4. 1 and 2 for incidents, 3 and 4 for static jobs, 1 being the highest priority. Some customers have more expensive fees for tardiness and thus have a higher priority.
- An availability time or occurrence time. All static jobs are available at $t = 0$. Incidents will become available during the day. The availability time of an incident is equal to its time window start time a_i, because incidents can always be visited as soon as they become available, in contrast to static jobs.

The jobs which are known a-priori will be referred to as *static jobs*. Static jobs have an average service time of 25 minutes, ranging from 1 minute for a short check to 8 hours for a surveillance. The dynamic jobs are referred to as *incidents*. Incidents have an average service time of 16 and a half minute, ranging from only a few seconds (false alarm) up to multiple hours in case of a burglary arrest. However, usually an incident takes 10 to 30 minutes. Locations are usually clustered due to business areas.

2.3 Relevance of this work

Pillac et al. [16] give us a non-exhaustive list of applications for vehicle routing problems where there usually is a dynamic component. These applications include but are not limited to: delivery of petroleum or gases, courier services,

inter modal services, combined pickup and delivery services and management of containers. Usually the current solution is one or multiple human coordinators or a clear separation between a-priori known jobs and dynamic jobs. (As in our business case.) An effective implementation of a DVRP solver could bring a great deal of efficiency in some businesses.

This research will also try to contribute to closing the gap between the large number of theoretical optimization algorithms and the practical implementation of these algorithms, as is also discussed in [9]. We will give some advice on implementing theoretical (dynamic routing) problems to use in a real-world situation.

3 Ant Colony Optimization

The Ant Colony Optimization (ACO) algorithm was introduced by [6], then called *the ant system*. ACO is a metaheuristic, which means it can be applied to a large variations of problems and across various domains. ACO is inspired by the way ants gather food. This process is shown in Figure 1. One or multiple ants will walk a relatively random path from their nest (N) towards a food source (F) and back. An ant will leave pheromones. Other ants will smell these pheromones, which will give them a higher probability to choose the same path. If there is more pheromone on a path, the probability that an ant will choose it will be higher. However the probability will never be 0, so ants can diverge from the pheromone-intensive paths, allowing for exploration. Eventually, the shortest path will be traversed the greatest number of times, since it takes the least amount of time to complete, thus making the pheromone count on that path higher. (As shown in Figure 1.3.)

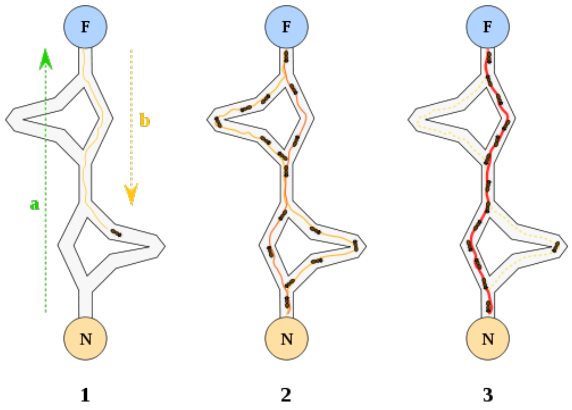

Figure 1: The creation of an ant colony path using pheromones. [7].

This food gathering method inspired the ACO algorithm, which uses the same pheromone based techniques to solve various problems, for instance routing problems. This section will describe how it can be used in a VRP and a DVRPTW.

3.1 Canonical ACO algorithm

The canonical ACO pseudo algorithm can be seen in Algorithm 1. Each line within a graph will have a pheromone level $\tau \geq 0$. One by one, each ant within the colony will find its target by taking one step at a time. For each step

the ant will look at the pheromones of the available nodes, which are usually neighboring nodes which the ant has not visited yet. It will choose its next step based on these pheromones, i.e. by roulette wheel selection [13]. When all ants have done this, the old pheromones will decrease in value (evaporation), whereas popular paths increase their τ. Eventually, the best found solution will get better with time.

Algorithm 1 The canonical ACO algorithm

```
1: Initialize pheromones
2: while not terminate do
3:     for every ant do
4:         while ant has not found goal do
5:             for each available neighbor node do
6:                 Calculate probability for node based of pheromones
7:             end for
8:             Take pseudo-random step towards a node
9:         end while
10:        Evaluate and save route
11:    end for
12:    Save best route T*
13:    Evaporate old pheromones
14:    Update pheromones
15: end while
```

3.2 ACO in VRP

Being a metaheuristic, ACO can be used to solve a variety of problems. Mostly it is used to solve graph-like discrete problems, like the VRP.

Bell et al. [1] describe how a VRP can be solved using ACO. To apply ACO to a VRP, the vehicles will need to be simulated as follows. An ant will start at the depot and select its first step by choosing a customer i, based on the pheromone levels between the available customers and the depot. After this the ant will select the new customer j in the same manner. Available customers can be all customers which have not been visited yet, or an arbitrary number of nearby customers, as seen from the ant's position. An ant continues constructing a tour until the moment that visiting a new customer would always violate a capacity constraint, or in our case a time window constraint. Then a new vehicle will be simulated. The ant will go back to the depot and start constructing routes by visiting the remaining customers within V. This will con-

tinue until all vehicles are used or all customers in V are visited. Pheromones can be distributed locally after each ant and globally after each colony, based on the best found solutions. In some ACO variants each ant represents a vehicle.

3.3 ACO in DVRPTW

Van Veen et al. [24] describe an ACO algorithm designed for a DVRPTW. This algorithm is an extension of the ant algorithm described in [10], which performs very well on Solomon's VRPTW benchmark problems [21] according to [24]. The DVRPTW of [24] will be summarized briefly.

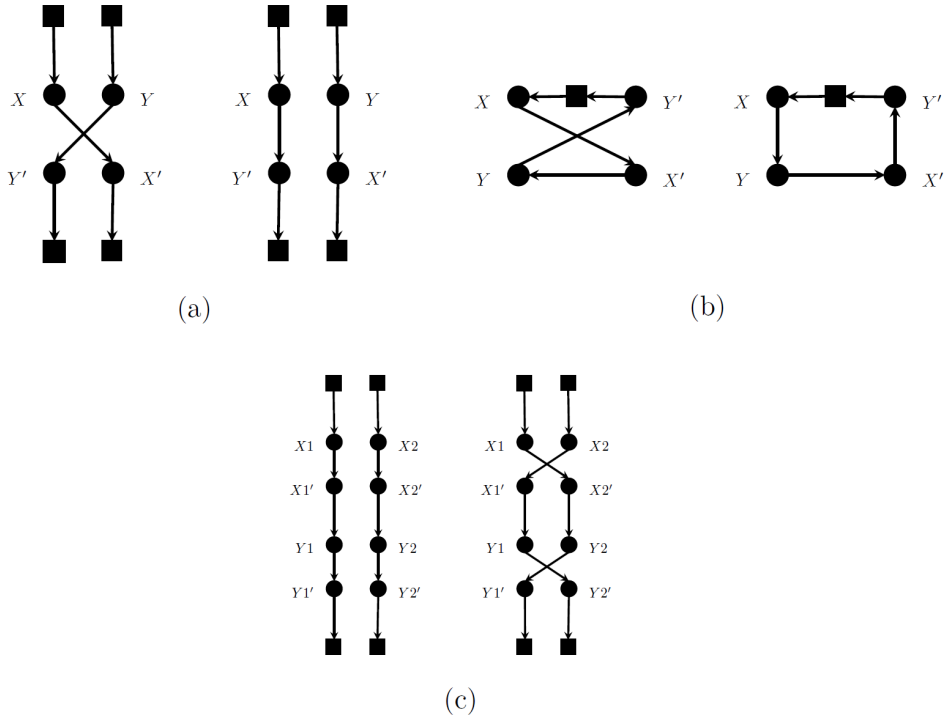

Figure 2: Examples of 2-opt edge replacements. Squares represent depots, circles represent nodes. (a) demonstrates a move with edges from different tours. (b) is an example of a move within a single tour. (c) shows the process of cross exchange. [24]

The algorithm is a *MACS-DVRPTW* algorithm. (Multiple Ant Colony System DVRPTW.) It uses two ant colonies to approach an optimal solution. The ACS-VEI colony searches for a solution with less vehicles than the best current solution (T^*). ACS-TIME searches for a solution with less total driving time/distance than T^*, while not changing the amount of vehicles. ACS-VEI works with infeasible routes, working towards a feasible route with less cars than T^*. ACS-TIME does not work with infeasible routes. Both colonies are started by a supervising system, referred to as the *controller*. The governing principle of the controller is running the colonies and processing any solutions from those colonies. "A solution with less vehicles is always preferred over a solution with a smaller distance." ACS-TIME also uses a technique called *cross-exchange*, which is presented in [22]. This local search method is used to optimize the routes further locally. It can be seen in Figure 2.

Algorithm 2 The controller of the MACS-DVRPTW algorithm in [24].

```
1: Set time t = 0
2: T* ← NearestNeighbor
3: while still time steps left do
4:     Start ACS-TIME with r = r_{T*}
5:     Start ACS-VEI with r = r_{T*} − 1
6:     while colonies are active and time step is not over do
7:         Wait until a better solution T is found
8:         if If r_T < r_{T*} then
9:             Stop colonies
10:        end if
11:        T* ← T
12:    end while
13:    if time-step is over then
14:        Stop colonies
15:        Update available nodes (incidents)
16:    end if
17: end while
```

The pseudo code of the basic outlines of the MACS-DVRPTW algorithm can be seen in Algorithm 2. To find initial solutions, the nearest neighbor heuristic [8] is used. This heuristic starts and continues expanding a route with the nearest available customer which isn't visited yet, until it reaches a constraint. Because this nearest neighbor heuristic also has a limit to the number of vehicles, it can give infeasible solutions. However, this is not necessarily an

unwelcome outcome, since this solution is only used as a base solution to improve upon by the ant colonies.

Using the nearest neighbor solution to initialize pheromones, both colonies are started and will start looking for a solution. ACS-TIME will look for a time-efficient solution with the current amount of vehicles, ACS-VEI will start looking for a solution with one less vehicle than the current solution. The dynamic customers are introduced at certain time-steps. (Where $t > 0$.) Each time a new customer is introduced, the colonies are stopped and started again including the new customer. When ACS-VEI finds a solution with less vehicles (r_T), both colonies are restarted also. (Line 8 of Algorithm 2.) Eventually, the controller will continue improving the routes while constantly receiving new orders at certain time steps. These new orders will then also be incorporated in the routes.

3.4 Other related work

More relevant research has been done on the topic of dynamic vehicle routing problems in combination with ACO. Some relevant works will be listed here.

Rizzoli et al. [18] discuss two real-world problems being solved with ACO: a VRPTW and a VRP with pickup and delivery. The ACO algorithms performed well on both VRP cases. DVRP was discussed in this paper, but not put to practice.

Pillac et al. [16] explain a number of algorithms designed to solve a DVRP, which can give useful insights when solving a DVRP. [16] includes an ACO algorithm which uses time slices. The problem execution is divided in a number of periods, or slices. The algorithm only recalculates the routes at the end of each slice.

Finally, the authors of the MACS-VRPTW algorithm [10] have published a paper on the successful implementation of two applications of ACO in a VRP. [11] One is based on the MACS-VRPTW algorithm, the other has a prototype element for DVRP handling, which has promising results so far. The algorithms are described in [11], but no sufficient details of the actual implementation are discussed.

4 Research Methods

This section will explain the methods used to conduct the research. First of all, Section 4.1 will explain the global ideas of our implementation and analysis processes, without going into too much details. After that, the details of our methods will be discussed step by step in Section 4.2.

4.1 Research approach

4.1.1 Implementation theory

The implementation of the algorithm on a logistics platform was achieved in an iterative process. A number of changing requirements was expected, as well as the rise of some unexpected complications during the development. Brodley et al. [2] present a model, based on the classic waterfall model from [19]. However, this model is more suitable for iterative processes compared to the waterfall model. The model in [2] is made for applying classification algorithms in practice, but is general enough to apply it to a VRP algorithm. It can be seen in Figure 3. Key in this model is the feedback coming from stages 3 and 4, testing and usage, respectively.

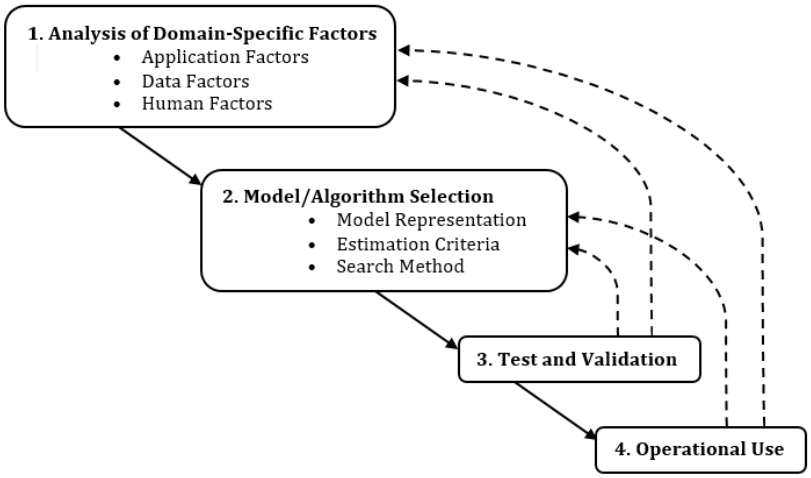

Figure 3: The application development process according to [2].

Also Hevner [12] presents a model for design and development within a scientific context. In Figure 4 we see his design science research cycles. In contrast to Figure 3 we can see that these cycles are based around certain domains. (Environment, Design Science Research, Knowledge Base.) The environment

of our research is the business case from Section 2.2 and the real-world testing. The knowledge base is existing and newly derived scientific knowledge and experience with respect to the ACO algorithm and the VRP. It also includes any newly scientific results and conclusions. The Design Science Research is everything in between. The three cycles links these three domains together and can be partly linked towards the arrows in Figure 3. During the implementation of the algorithm it was always helpful to consider what cycle the implementation was going through at a certain moment. This way there was a clear separation between theoretical analysis and real-world validation.

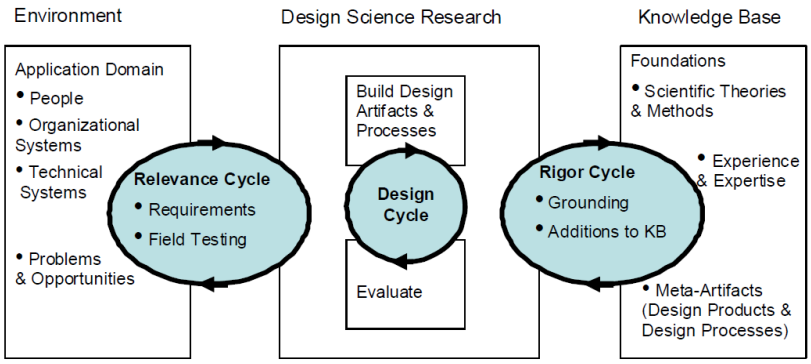

Figure 4: The Design Science Research Cycles from [12].

4.1.2 Global Research Processes

In this subsection the global processes of the research will be explained. With this we mean the overall strategies and concepts used to produce the more detailed techniques of the research.

Consider the development process of Figure 3. It shows us that the first stage is the *analysis of domain-specific factors*. First of all, data was gathered so the case of Section 2.2 could be made. An interview with the company's data analyst was conducted and customer data (job times, locations etc.) was gathered. This was done to gather initial requirements and get a grip on the situation. (The Relevance Cycle.)

After the first analysis, the actual design and implementation started. An initial set-up for the algorithm was designed and implemented. We would go back and forth between implementing and re-designing. (stage 2 and 3 of Figure

3 and the Design Cycle of Figure 4.) The real-world testing is considered as the Operational Use stage, this is the 'Field testing' part of the Relevance Cycle. Any unexpected trial data derived from the real-world tests could be used to contribute to the Knowledge base. (The Rigor Cycle.) For more detail on the implementations and real-world testing, see Section 4.2.

4.2 Implementation and trial set-up

This subsection will explain the details of the research. First the test case which was used for the trials will be discussed. Then the method of the initial implementation is explained. The execution of both real-world trials will be discussed, including the intermediate revisions. Finally, the surveys held after the trials will be shortly discussed.

4.2.1 Test case

To use our business case as a practical real-world testing case for a DVRPTW algorithm, the case needed to be scaled down. For 400 incidents a few dozens of vehicles would be needed. A trial of this size would be outside of our scope, because of finances, time and complexity. Therefore, a testing case of 5 vehicles was created. 4 static job vehicles from the same depot and day and with similar addresses were selected. These 4 vehicles have a total of 48 jobs. Also, one incident vehicle from the same area and day was selected, containing 9 incidents. This gives us a dynamicity of 15.8%, $(9/(48+9))$ which is relatively high compared to the average of 11.6% in the business case in Section 2.2. This was done on purpose to give a challenging test case. The 57 orders were made anonymous by selecting an address zero up to two streets away from the initial address. This was done to keep the customers anonymous, while still maintaining a good test case. The time windows of the jobs within the test case all took place within a 6 hour time-frame, in the evening. To get a general view of the addresses in the test case, the map with all customers can be seen in Figure 5.

4.2.2 Initial implementation

To successfully implement a DVRP it is key to know the location of the vehicles' location and status at the moment of occurrence of a new job. To achieve this, the DEAL platform [14] was used. This platform is made for logistic solutions such as the efficient delivery of packages. All drivers can use a mobile application to update their status and GPS location. The DEAL mobile application also shows the sequence of jobs and its addresses to the drivers and to the coordinators. The ACO algorithm was implemented on an external algorithm

Figure 5: All test case jobs on the map. Blue = static jobs. Red = incidents.

agent which could get an overview of the available jobs and the vehicles. When triggered, this algorithm agent used ACO to rearrange the routes of the vehicles.

The first step was to implement the MACS-VRPTW of [24] for the agent so it could find efficient routes when given a set of jobs. Apart from this algorithm, some more features were necessary to fit the business case problem. One by one these additional requirements were added and tested. The service time (s_i) and priorities (p_i) were the first functionalities to be completed and tested. Most functionality testing was performed on a small number of real-world jobs, enough to fill the time of 2 or 3 cars, because testing on the full test case made the tests too complex and time consuming. Initially there was no dynamicity implemented, since the ACO algorithm had to function properly before advancing to this stage, to avoid unnecessary complexity. When the ACO algorithm for the a-priori calculations was verified, an extra trigger method was created so the agent could be called each time a new incident became available.

The following features were implemented during the initial implementation, in the following order:

1. A regular ACO metaheuristic.
2. The extension of the ACO algorithm to fit a VRPTW, including the MACS controller.

3. A service time constraint s_i for each job.

4. Job priorities p_i. A job with a higher priority has a higher chance to be included into a solution provided by an ant colony.

5. A method to include a new job (incident) utilizing local search, including cross-exchange. This method could be triggered as an event as soon as an incident became available.

1, 2, 3, and 5 could be taken almost directly from the algorithm of [24], although the original algorithm did not need a trigger to add an incident. A trigger was more practical for our real-world situation, because a coordinator can activate it manually.

4.2.3 Trial 1

To test how well the algorithm performed in practice, two trials were conducted. The first trial consisted of two teams with 5 drivers each. As Section 4.2.1 discussed, our test case was made for 5 vehicles. Team A tested the performance of our algorithm. Team B worked according to the solution of the baseline algorithm provided by the security company: here 4 cars were assigned to static orders in a predetermined schedule, while one car visited all the incidents. Team B was used as a control group for baseline comparison. This way it was possible to see if there actually were improvements.

All drivers were students or recently graduated students. The advantage of working with these people was that they were accepting to contribute to the project for a relative low salary and their recruitment was relatively easy due to the university setting of the project. Also, students proved to have flexible working hours. A survey was conducted inquiring the amount of experience of the drivers. Four out of ten drivers had experience with delivery by car and six had more than 4 years of driving experience. Each driver drove regularly. (Ranging from 3 times a month to every day.) The most experienced drivers were distributed over both teams. The relevant experience of the drivers can be viewed in Table 1.

The initial routes for team A were calculated beforehand by the ACO agent. Team B's routes were the same as the security company's predetermined routes. Both teams got their jobs assigned to them through the DEAL mobile application, to get a fair comparison between teams. However, team B's incident driver got a text message each time he was assigned to a new incident, just like in the original case. Team A's drivers were instructed to be aware of changing

Driver #	Trial 1 team	Trial 2 team	Driving experience (years)	Driving frequency (days per week)	Navigation through Rotterdam frequency (days per week)	Delivery experience (by car)
1	A	C	> 8	1-2	1-2	Yes
2	A	-	5-8	6-7	< 1	Yes
3	A	-	4-5	3-4	1-2	-
4	A	C	3-4	6-7	3-4	Yes
5	A	-	2-3	3-4	1-2	-
6	B	C	5-8	3-4	< 1	-
7	B	-	5-8	6-7	1-2	Yes
8	B	C	4-5	3-4	3-4	-
9	B	-	2-3	6-7	6-7	-
10	B	-	2-3	< 1	< 1	-
11	-	C	4-5	1-2	< 1	Yes

Table 1: The relevant experience of each driver.

routes at all times. Each time an incident became available, the agent was triggered to change team A's routes, this was done on-the-fly.

Team B started at predetermined times, determined by the time it would take them to reach their first address on time, according to the security company's planning. Team A's vehicles all were available from the start, since there was a possibility one of them was assigned to an incident.

4.2.4 Revisions

The conclusions drawn from the first trial (discussed in detail in Section 5) were used to improve the implementation of the algorithm. A list was made of each required improvement and these were implemented iteratively. The most important revisions were:

- Balancing of the vehicles. During the trial some vehicles were very busy, while others had hardly any work. (I.e. 25 and 2 jobs respectively.) This can be seen in the results section, (Section 5) where Figure 6 shows a vehicle with a significantly high amount of vehicles during the entire trial. This fact resulted in the busy vehicles being late. Balancing also helps to

give some buffer time, in case an incident has to be handled. Balancing was achieved by giving the vehicles a maximum amount of orders during the nearest neighbor algorithm. This maximum equals: $n/(r-1)$.

- When a driver is already performing a job or driving towards a job, he/she should not be interrupted. I.e. this job should not be reassigned to another driver.
- At the moment of recalculating the routes, it is important to keep track of the current time and the current position of the vehicles to check if any vehicles will be late. It might be necessary to reschedule to prevent tardiness.
- The vehicle speed used in planning was assumed too high initially, since most of the trial took place in a urban area. It was reduced from 80 km/h to 30 km/h.

4.2.5 Trial 2

The second trial consisted of only 5 drivers, referred to as team C. For practical reasons another control group was not included. The first control group results proved very consistent and there was no strong need to test these results again, since the situation was very similar. Both trials were conducted on a Friday, during the same time period, with no large weather differences, etc. This choice also allowed us to focus more on the algorithm's results and cope with any algorithm problems during the trial.

Much like team A of Trial 1, the 5 cars of team C were sent out to visit their dynamic routes, which were determined on-the-fly by the (improved) algorithm agent. This time, there was a bigger focus on the minimization of labor hours, therefore not all cars started at the beginning of the trial. Two cars started driving at the start of the trial, two were given a custom starting time, based on the start of the time window of their first planned job, accounting for driving time to get there at the start of the time window. The remaining car was on stand-by, in case the two active cars did not have enough buffer time to pick up an incident.

Some decisions were human made during this trial, but could theoretically be adopted into the algorithm. For instance, the start times of the vehicles were not conceived by the algorithm. This was not the only human intervention during the trial. When the algorithm saw a route as infeasible, it could drop one or more jobs to get a feasible solution. This happened with 5 orders.

However, 3 of these orders later became feasible again and were assigned to a driver manually. This intervention was necessary because the algorithm was not designed to pick up unassigned orders again.

During this trial the drivers took some forms with them so they could take notes about their jobs, including arrival times and stress-levels. This was done to gather any interesting insights in the human factor of the implementation. An example of these forms can be found in the appendix.

4.2.6 Surveys

As can be seen in the first case study of [18], sometimes a problem solution may seem very efficient, but doesn't work or isn't ideal because of the people involved. Therefore it is important to also inquire the participant's experiences. At the end of both trials all drivers did a survey to rate their experience. This survey inquired about factors such as stress and certainty, but was also made to identify any problems during the trials. The same survey was used for both trials. All qualitative results of this survey were processed using *grounded theory coding*, as shortly described in [4]. This process goes roughly as follows:

1. *Open coding.* Labels are created based on the rough data. Labels should summarize the most important discussions of the test subjects, in this case the drivers. Labels are then grouped together based on subject.

2. *Axial coding.* Connections between the open codes are identified. Labels are grouped together more formally.

3. *Selective coding.* Based on the axial codes, the most important conclusion or conclusions should be drawn. The data will be examined again, while looking for more examples which can strengthen these conclusions.

5 Results

This section contains all results of both conducted trials. First of all, the timing of the vehicles' routes will be discussed. An important factor is the exact timing of all job executions. After that, the human factor will be addressed, such as the survey results. Finally, the new and final implementation of the ACO algorithm will be discussed since this can be viewed as an important final result as well.

5.1 Timing performance

The MACS-DVRPTW algorithm was implemented and tested as described earlier. The DEAL platform [14] allowed us to record the relevant data to draw conclusions from these trials. After both trials it was possible to see when jobs were started and finished. This gave good insight to the timing of the algorithm.

Static jobs			
	Control group	Trial 1	Trial 2
Not visited (#)	0	16	2
Not visited (% of total)	0	33.33	4.17
Late (#)	0	6	5*
Late (% of finished)	0	18.75	10.87*
Late (minutes)	0	106	50*
Too early (#)	1	0	1
Too early (minutes)	8	0	3

* 3/5 were caused by a traffic jam, causing 44/50 late minutes.

Table 2: The timeliness of the 48 static jobs for both trials.

Incidents			
	Control group	Trial 1	Trial 2
Not visited (#)	0	4	0
Not visited (% of total)	0	44.44	0
Late (#)	0	1	1
Late (% of finished)	0	20	11.11
Late (minutes)	0	57	2

Table 3: The timeliness of the 9 incidents for both trials.

For our algorithm to perform well on the business case it is important that there are as little contract violations as possible. Therefore, it is important to look at the timeliness of drivers, since they could arrive too late. It is also possible that a job is not visited at all, either because the driver was running too late or because the algorithm saw this as infeasible. In a very rare occasion (twice) the job was started before the time window, this is (in our case) due to human error.

In Table 2 these results regarding the static jobs are shown. The incident results can be seen in Table 3. These results show us that the control group performed relatively well and stable. No control group driver arrived too late for either a static event nor for an incident.

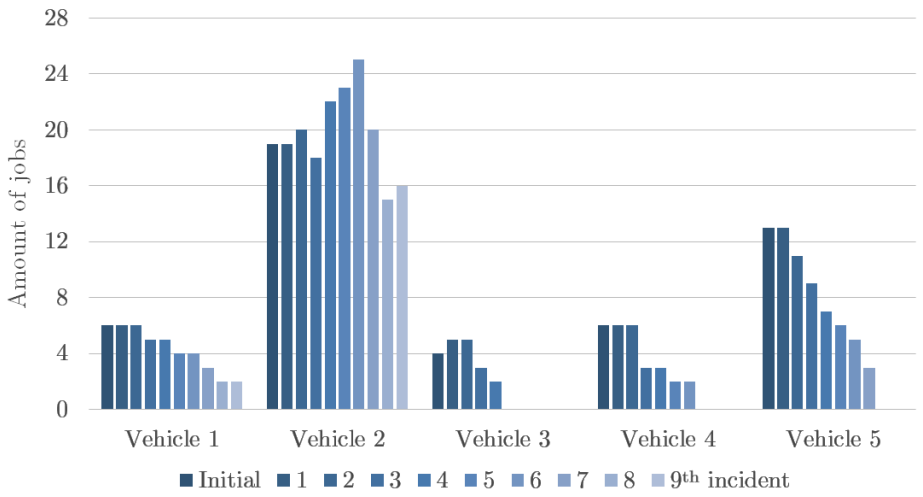

Figure 6: The total amount of jobs during Trial 1 in the vehicles' schedules, for each incident availability time.

The first algorithm trial experienced some problems. The most important problems are mentioned in Section 4.2.4, since they were used to improve the implementation before Trial 2. The problems in Trial 1 caused a significant amount of jobs to fail or at least be late. This can be seen in both Table 2 and 3. More than one third of the jobs were not finished in Trial 1. This is not acceptable for the business case. An important cause of this tardiness was the fact that one vehicle was scheduled to have more jobs than it could handle. In Figure 6 it is visible that vehicle 2 was given a lot more orders than the other

vehicles. This problem remained during the entire trial, even though vehicle 3 was already finished with its jobs by the time the fifth incident occurred. This vehicle could have taken on some of the excess jobs from vehicle 2.

After making the improvements of Section 4.2.4, Trial 2 was conducted. A great improvement since Trial 1 can be observed. In Figure 7 it is visible that the total amount of jobs is more evenly distributed between vehicles and that these total amounts have a downwards slope as time progresses.

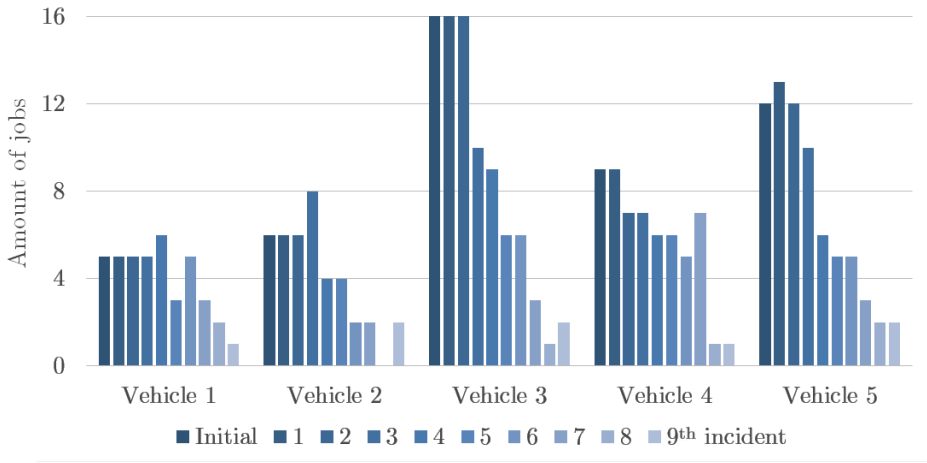

Figure 7: The total amount of jobs during Trial 2 in the vehicles' schedules, for each incident availability time.

Partly because of this even distribution, the timeliness of Trial 2 was a lot more acceptable. Only 2 (static) jobs remained unvisited. 5 jobs were too late with a total late time of 50 minutes. However, halfway through the trial, one of the drivers got stuck in a traffic jam which was not at all present during the control group trial. 2 jobs were in the middle of this traffic jam, both with an arrival time relatively close to the planned arrival times of the control group. (Within the same hour.) So it is safe to say that the control group would also could have experienced some delay. Or at the very least we could say that the Trial 2 driver would have experienced less or no delay if the traffic jam would have not been present. Therefore, if the traffic jam wouldn't have been there, Trial 2 would have had only 2 late orders, with a total of 6 late-minutes, making these results a lot more positive. However, there is no way of knowing this with complete certainty.

As mentioned in Section 2.2.1 the most important metric is the total labor time. These results are presented in Table 4. The total labor time needed would be the accumulated driving times of all cars, including driving from and towards the depot. The driving-times without depots are also shown to give an impression of the performance excluding the startup and completion of the trials. The total time of Trial 1 seems to be the shortest, but this is because so many jobs were left unfinished. For Trial 2 we see an increment in total labor time of 8.6%.

Total driving time in hours			
	Control group	Trial 1	Trial 2
Excluding depot	25:59	19:57	22:57
Including depot	27:26	21:33	25:05

Table 4: The total driving times, or labor hours, for both trials.

5.2 Driver experience survey results

5.2.1 Trial 1

The most important outcomes of the survey of Trial 1 were:

1. The changing of routes was experienced as 'confusing' by some drivers.
2. Some jobs disappeared while a driver was executing it. This raised uncertainty.
3. A driver felt 'a bit useless' since he had to drive back and forth from one side of the city to another side and back again. Even though this might be the most efficient route, a driver might not experience it as such.
4. Most stress was experienced when a driver was running late.
5. Most users felt more certain because they got a clear explanation beforehand and because they could contact a coordinator at all times.
6. There were some user interface problems, i.e. some users marked a job as started by accident and this could not be reversed by the user.
7. Most drivers felt the planning was tight, but not too tight or stressful.

These last outcomes (4, 5, 6, 7) were also found in the surveys of the control group drivers. Outcomes 1, 2 and 3 were only relevant for the drivers that tested the dynamic ACO algorithm. (Team A.)

5.2.2 Trial 2

From the survey of Trial 2, the same outcomes as number 4, 5, 6 and 7 of Trial 1 were found. Furthermore, the following results came out of the survey:

- Two drivers found that a more frequent refresh of the job list would be helpful. A forced refresh each time a route is changed might even be more effective.
- One driver experienced quite some stress during a traffic jam.
- Four drivers already participated in the first trial, and experienced the second went much smoother. This was accounted mostly towards the relative absence of problems, such as disappearing jobs.

5.3 Forms

As mentioned in Section 4.2.5, the drivers of Trial 2 were given a form to write down their arrival times and also their stress and confidence or certainty level, ranging from 1 to 5, were 1 is '(almost) none' en 5 is 'a lot'. They rated their stress and confidence when arriving at a job.

At most times (42/55) stress was 1 and confidence was 5. When stress went up, that usually meant that the driver's confidence was low. (7/12) The drivers experienced stress in the following occasions:

- The driver was running late.
- The driver got stuck in traffic.
- The driver took a wrong turn, delaying his route.
- The driver was not sure if finishing a job outside of the time window also counted as being late.

5.4 Final implementation

One of our most important results is the actual final implementation of the algorithm. The theoretical algorithm from Section 3.3 was adjusted so it could be used in a real-world situation. This version of the algorithm will be described in this section.

The controller of the implemented algorithm is very globally displayed in Algorithm 3. The adjustments to this controller are:

- The algorithm is not constantly searching for better routes. This is because the amount of changes to driver schedules should be minimized to avoid confusing the drivers. The cost of a small change would possibly be greater than its gain. Line 16 in Algorithm 3 shows that the algorithm is not actively calculating after updating the schedules and before a new incident is introduced.
- The number of iterations used by the ant colonies was set to 5000. (Line 26 in Algorithm 3.) This number was found to produce acceptable results within a minute. A short total calculation time was necessary to update routes as quickly as possible after an incident occurred. Note that this number might need to be changed when the test case is scaled up or down.
- The first job of a vehicle will always be locked on the first position of its route, as is seen in line 17 of Algorithm 3. This is so the driver never loses a job he/she is already performing. Also, when a driver started driving towards a customer, this customer should not be rescheduled to another driver.

Other important adjustments to the algorithm were:

- If the colony is trying to add missing nodes to an infeasible route, the highest priorities will be added first, if possible. The missing nodes from line 18 of Algorithm 3 are sorted by priority.
- Feasibility of a route is based on the current location of the vehicles, which can be viewed as starting positions or depots when introducing an incident. Feasibility is also based on the time at the moment of calculation. Therefore, past time windows will not be considered anymore. By considering time and vehicle locations, more accurate schedules can be made when introducing a new incident while vehicles are driving towards a job. The feasibility check on line 23 of Algorithm 3 is based on the time and location which are retrieved on line 22.
- Driving speed is by default 30km/h, which is a good average for urban areas, allowing for some buffer time.
- The nearest neighbor heuristic on line 2 of Algorithm 3 intents to distribute the customers relatively evenly over the vehicles. This will give a balanced initial solution for the ACO pheromone initialization. This even distribution within the nearest neighbor algorithm is achieved by giving each vehicle a maximum of $n/(r-1)$ customers.

Algorithm 3 The controller of the final implementation of the MACS-DVRPTW algorithm.

```
1: Set time t = 0
2: T* ← NearestNeighbor
3: while not terminate initial calculation do
4:     Start ACS-TIME with r = r_{T*}
5:     Start ACS-VEI with r = r_{T*} − 1
6:     Wait until a solution T is found
7:     if If r_T < r_{T*} then
8:         Stop colonies
9:     end if
10:    T* ← T
11: end while
12: Stop colonies
13: Update routes
14: Start execution of problem solution
15: while execution of DVRPTW is not over do
16:    Wait for new incident
17:    Lock current task of each vehicle
18:    for each missing node do
19:        Calculate cost of each possible insertion in each route in T*
20:        Insert node were cost is lowest
21:    end for
22:    Get current time and vehicle locations
23:    if routes are infeasible then
24:        Start ACS-TIME with r = r_{T*}
25:        Start ACS-VEI with r = r_{T*} − 1
26:        while stop condition is not reached do
27:            Wait until a solution T is found
28:            if If r_T < r_{T*} then
29:                Stop colonies
30:            end if
31:            T* ← T
32:        end while
33:        Stop colonies
34:    end if
35:    Update routes
36: end while
```

6 Discussion and Conclusion

6.1 Discussion

This section will first discuss the performance of the algorithm in practice, mainly looking at the second trial. We consider what scaling up the test case could do for these results and the findings during the implementation will be briefly considered. The end users, mostly the drivers, will also be discussed.

6.1.1 Performance of the implementation

To discuss the performance of the implementation of the MACS-DVRPTW algorithm, the timing results of Section 5.1 are an important factor. As can be observed, the results of the first trial are unacceptable when compared to the control group's results. However, we are most interested in the performance of the final implementation, therefore the results of Trial 2 are much more relevant.

Firstly, it is visible that the result for the incidents are very acceptable. Only one incident was late, only 2 minutes. This is a relatively small constraint violation and might even be overlooked in practice. The results for the static jobs, however, are not near those of the control group. Two jobs did not get finished at all, which should not happen when we know it is theoretically possible to visit all of them. (As the control group did.) 5 jobs were late with a total of 50 late-minutes. In comparison with the control group this might look like a significant deterioration. However, if the traffic jam (which was also mentioned in Section 5.1) is taken into account, Trial 2 would have had only 2 minutes and 6 late-minutes. In practice such small delays might be overlooked. However, we cannot be completely sure it would have happened like this if the traffic jam had not occurred.

Our main target was to minimize the total labor time. The total driving times of Team A of Trial 1 can be disregarded. Trial 1 has the lowest total amount of driving time, however this is not a realistic value, since 33% of the static jobs and 44% of all incidents were not visited. The total driving time of Trial 2 is 2 hours and 21 minutes less than the driving time of the control group. This is most probably a lot more than needed to visit the two missing jobs, which have a total execution time of only 19 minutes. Therefore it can be said that the algorithm could reduce the total amount of labor hours by a significant amount of hours.

These results should be observed critically. Although the improvement in labor time seems significant, it is only a single test case. The test case was

made as representative as possible, but with just these results we cannot be completely sure it will always shorten the total labor time. However, if further improvements are made towards the algorithm (see Section 7.1) it might be possible to avoid any failed or late orders.

6.1.2 Scaling up the test case

Our test case was based on the data of 4 static job cars and 1 car handling incidents. We could consider what happens when this test case is scaled up. A possibility would be to use all the cars within the Rotterdam area to test the MACS-DVRPTW algorithm. In our 5 vehicle test case, we would usually have 1 to 3 vehicles nearby when an incident became available. But when there are around 20 vehicles in this area, the number of available nearby vehicles will grow, as will the possibilities to rearrange any routes when the incident delays a vehicle. This can make the overall performance (measured in labor time and fines) better, i.e. because less stand-by vehicles are needed and planning can be tighter. It can also decrease the tardiness, making the algorithm more robust.

6.1.3 Implementation findings

Some difficulties were faced during the implementation process of the algorithm. Out of these difficulties it is possible to draw conclusions for any further research within the field. This can also be said for things that went well.

Most problems occurred within the Relevance Cycle. (Of Figure 4.) We found that it is hard to test a practical implementation without the actual resources. I.e. the algorithm had to be tested throughout the implementation period before the first trial. It was not completely clear what to test and how to test it before running the first trial. This is mainly why the results of this trial were unacceptable and a second trial was needed.

Since expertise was divided across project team members, the communication sometimes was a challenge. Especially when the algorithm agent had to connect to the DEAL platform in a correct manner, a combination of expertises was needed. Therefore it was important to establish clear roles.

6.1.4 Driver findings

When implementing a real-world application it is important to consider the people involved, mostly the end-users. In this case the end-users involved were the drivers. The most important finding regarding the driver experiences came

from the survey results. (Section 5.2.)

Most stress was experienced when a driver was running late. One of the algorithm's tasks is to prevent this, so this stress-factor will decrease once the algorithm will improve.

Another important factor was uncertainty. The changing of routes was found confusing by some drivers. This element can of course not be changed within a DVRP. However, a valid approach is keeping the amount of changes to a minimum. It is also important to make the changes clear by giving a good overview of finished and future jobs. It is important that the driver knows what is expected of him/her. Therefore it is also important that it is made clear when a driver's route changes. These things can be achieved through the user interface on their mobile device. Another important factor contributing to certainty is a clear briefing.

Another interesting finding was about the usefulness the users experienced. Driving back and forth between city sections might seem the most efficient for an algorithm, but a driver might perceive it as unnecessary. This could be fixed by preventing such situations in the algorithm, or giving the driver a clear indication of his/her usefulness, i.e. by being transparent about any routing choices, for instance by clearly displaying the time windows or even by explaining unconventional strategies.

6.2 Conclusion

6.2.1 Research question 1

The first research question was: *How can the algorithm of [24] be successfully put to practice?* In this document, the process of the implementation of the algorithm was described. In Section 5.4 we can also see the adjusted algorithm for practical use, our end result.

As discussed in the discussion, our implementation gave relatively good results on our (second) real-world trial. The total labor time was improved significantly, with good promise for more improvements. A small number of jobs failed or were started too late, but we expect this can be fixed after some minor adjustments to the implementation. (See Section 7.1.) Therefore it can be concluded that we have described a good basis for the practical implementation of the MACS-DVRPTW algorithm of [24].

6.2.2 Research question 2

The second research question is more focused on the general concept of putting theory into practice. The question was: *What should be considered when implementing a (dynamic routing) algorithm for a real-world environment?* A number of general lessons learned and new insights can be compiled to answer this question:

- *Implementing practice means testing in practice.* When working with real-world cases and data, one cannot simply implement something and only test in theoretically. Multiple real-world tests are often required to identify the real-world problems of an implemented algorithm.

- *Working iteratively leads to success.* It is impossible to know all the functionality of the algorithm implementation beforehand. Therefore it is important to keep in mind that requirements might change. Especially a real-world test can give a clearer look on the elements needed. This does however not mean that it is not a good idea to get a good head-start on the requirements. Starting with a thorough analysis of the business case can give a good indication of what might need the most attention.

- *Communication is key.* Implementing an algorithm in a real-world environment is not a one-man-job. In our case we needed at least an algorithm, a logistics system (DEAL) and a business case. These elements had to be brought together by combining these expertises.

- *People are important.* The end-users should play an important role in the development of the end result. After all, they will be using it and if they don't understand the product that might influence the performance of the algorithm negatively. We found that a clear briefing and description of tasks and expectations contributed to the confidence of the drivers. Also, the changing of routes should be presented as transparent as possible so the employee comprehends the logic of his route to a certain extend, i.e. does not doubt the efficiency of the schedule. It is also important to consider that an employee needs to feel useful and needs to have the feeling that he/she is treated fairly.

7 Future work

The implementation of the MACS-DVRPTW algorithm is not entirely ready to use yet. For instance, some orders were dropped by the algorithm while this is not acceptable in our business case. Some minor adjustments still have to be made for it to work reliably on the work-floor. Some extensions to the algorithm will be introduced which should make it work outside of our controlled test environment by making it more flexible, automatic and robust.

7.1 Algorithm extensions

7.1.1 Soft time windows

The MACS-DVRPTW is designed to fit all jobs within their respective (strict) time windows. If this is not possible, it will come with a solution which does not include all jobs. However, in a lot of real-world cases, these time windows are not that strict. The same goes for certain (low priority) customers in the business case of Section 2.2. Calvete et al. [3] discuss a method to work with soft time windows. This will give a penalty to being a few minutes late or early, without completely dismissing the job. In our trials it was visible that giving a small amount of slack would have made the results more acceptable. I.e. being 2 minutes late on a job is much better than not visiting the customer at all, because not showing up will give a much higher fine. Soft time windows will give the algorithm more robustness.

7.1.2 Picking up jobs

In the current implementation of the algorithm sometimes a job got dropped out of the solution because of infeasibility. Once a job was dropped, the algorithm did not include it in the problem set anymore. It had to be added again manually. If these dropped orders were considered into the problem set again each time the algorithm recalculated a route, this would make the algorithm more automatic, thus it would need less human assistance.

7.1.3 Automatic incidents

In our business case, incidents will come in automatically via the customers. After this, the coordinator assigns them to a vehicle. In the case of the algorithm, the coordinator makes an incident job and passes it through to the algorithm. A system to pass incoming incidents directly trough to the algorithm agent would greatly improve the efficiency.

7.1.4 Long idle times

Our tests proved that in most cases the planning will be quite tight. However, sometimes a driver will have to wait some time before he can start to perform his job, because he arrived before the start of the time window. Say a driver arrives an hour before his next job's time window. He/she should be able to use this time to visit an incident. With the current implementation, his/her first job in line is always locked, so the driver won't be free to do this. In this very rare case this should not happen.

7.1.5 Surveillance

Some jobs in our business case (not in the test case) were 6 hour surveillance jobs. When a driver is performing this job, an incident might come up very nearby. The algorithm should be able to see that this 6 hour surveillance could possibly be interrupted for a moment to send this vehicle to the incident.

A method to solve this was thought of during the initial implementation phase. However, we chose not to use it in our final tests since it proved too complex and made for an un-representative test-case. The implementation would have taken a lot of time, while the total fragment of surveillance jobs longer than one hour was relatively small. (1.9% of static jobs.) Say we have a multiple-hour splittable surveillance job with a time window $[a_i, b_i]$ and a nearby incident. (Nearby can be determined by an arbitrary distance.) The surveillance job was spit up into two separate jobs i and j. The new time windows are $[a_i, b_i]$ and $[a_j, b_j]$, where b_i and a_j are set to the current time. The ACO algorithm will decide automatically if it is necessary for the surveillance vehicle to interrupt its current job.

7.1.6 Predict incidents

If the algorithm could somehow predict upcoming incidents, it could anticipate them. For instance: some patterns could be recognized. In Figure 5 we see clusters of customers. There's a higher probability that an alarm will go off in that area. It is also not uncommon for an alarm to go off twice at the same customer. These basic patterns could be used to anticipate incidents to create a larger buffer in a vehicle's planning. Solomon [20] uses *expert systems* and *artificial intelligence* to anticipate dynamic data. *Timeweaver* from Weiss et al. [25] or a similar method could also be considered to predict incidents. Gambardella et al. [11] use job prediction in combination with an ACO DVRP algorithm.

7.1.7 Predict traffic

Most time windows in our business case are around the morning and evening rush hours. This is probably also the case for some other businesses which could use a DVRPTW algorithm. Therefore it can be very relevant to recognize certain patterns in the traffic flow either to predict travel time or to avoid traffic jams or other unnecessary delays. For this purpose the algorithm could be given current traffic information. (I.e. from a (web)service.) Or predictions could be made beforehand, as is done in [15]. This could possibly have decreased the delays in Trial 2.

7.2 Other

7.2.1 User Interface

As was clear in Section 5.2 there were still some comments about the user interface. This could be investigated a bit more, to make the execution of the dynamic routes as effortless as possible.

7.2.2 Real-world testing

The algorithm was mainly tested on our test case. Although this is a relatively representative test case, the algorithm might perform differently on other cases. The implementation of the algorithm is not complete until it is tested on a variety of realistic test cases, using real employees in stead of hired students. Only then one can know if the implementation was a success and if the algorithm is robust enough to perform on a variety of problems.

Nomenclature

Abbreviations

VRP	Vehicle Routing Problem
TSP	Traveling Salesman Problem
CVRP	Capacitated Vehicle Routing Problem
DVRP	Dynamic Vehicle Routing Problem
VRPTW	Vehicle Routing Problem With Time Windows
DVRPTW	Dynamic Vehicle Routing Problem with Time Windows
ACO	Ant Colony Optimization
ACS	Ant Colony System
MACS	Multiple Ant Colony System

Concepts

Job	A customer or address which needs to be visited
Static job	A job which is known a-priori
Incident	A job which cannot be predicted and will become available during the execution of the DVRP
Dynamicity	The fraction of the DVRP customers which are not available at $t = 0$
Service time	The time it takes to finish a job while on-site
Availability time	The time an incident occurs and becomes available to the algorithm.
ACS-TIME	The ACS to minimize the driving time
ACS-VEI	The ACS to minimize the total amount of vehicles

Mathematical symbols and variables

r	The total number of vehicles
n	The total number of customers
c_{ij}	The nonnegative cost for going from customer i to customer j
d_{ij}	The nonnegative travel distance (or time) between customers i and j
$[a_i, b_i]$	The time window of customer i
s_i	The non-negative service time of customer i
V	A graph with all the VRP customers. $V = 0, ..., n$
G	The complete graph where $G = (V, A)$, where A is the arc set
D	The dynamicity of a DVRP. $0 \leq D \leq 1$
p	The priority of a job
t	The time variable of the DVRP
$T*$	The best found solution by the MACS algorithm
τ	The pheromone level of a graph line

Acknowledgments

Firstly I would like to thank Zhiwei Yang for his great work on implementing the MACS-DVRPTW algorithm on the algorithm agent. I would also like to thank Rick van Krevelen for all his help, both technical support and research advice throughout the entire project. Richard van Klaveren also played a big role during the implementation of the algorithm. And I would like to thank everyone else at Almende B.V. who helped me during my research.

Furthermore I would like to give thanks to my supervisor Michael Emmerich for guiding me through the research and I would also like to thank Thomas Bäck for being my second supervisor.

Additionally I would like to thank Mark for proofreading this thesis and finally I would like to thank Josephine for supporting me.

References

[1] Bell, J.E., and McMullen, P.R. 2004. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics, 18* (1), 41-48.

[2] Brodley, C.E., and Smyth, P. 1997. Applying classification algorithms in practice. *Statistics and Computing, 7* (1), 45-56.

[3] Calvete, H.I., Galé, C., Oliveros, M. and Sánchez-Valverde, B. 2007. A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research, 177* (3), 1720-1733.

[4] Corbin, J.M., and Strauss, A. 1990. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology, 13* (1), 3-21.

[5] Dantzig, G.B., and Ramser, J.H. 1959. The truck dispatching problem. *Management science, 6* (1), 80-91.

[6] Dorigo, M., Maniezzo, V. and Colorni, A. 1996. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 26* (1), 29-41.

[7] Dréo, J. 2006. ACO Branches. (Digital image.) *Wikipedia images*, Retrieved July 9, 2014, from Wikimedia: http://commons.wikimedia.org/wiki/File:Aco_branches.svg.

[8] Flood, M.M. 1956. The traveling-salesman problem. *Operations Research, 4* (1), 61-75.

[9] Fu, M.C. 2002. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing, 14* (3), 192-215.

[10] Gambardella, L.M., Taillard, E. and Agazzi, G. 1999. MACS-VRPTW: A multiple colony system for vehicle routing problems with time windows. in Corne, D., Dorigo, M. and Glover, F. ed. *New ideas in optimization*, McGraw-Hill, Maidenhead, UK, 63-76.

[11] Gambardella, L.M., Rizzoli, A.E., Oliverio, F., Casagrande, N., Donati, A.V., Montemanni, R., and Lucibello, E. Ant Colony Optimization for vehicle routing in advanced logistics systems. in *Proceedings of the International Workshop on Modelling and Applied Simulation* (Bergeggi, Italy, 2003), ACM, New York, USA, 3–9

[12] Hevner, A.R. 2007. A three cycle view of design science research. *Scandinavian journal of information systems, 19* (2), 87-92.

[13] Lipowski, A., and Lipowska, D. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications, 391* (6), 2193-2196.

[14] Mahr, T., and de Weerdt, M. Distributed Agent Platform for advanced logistics. in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems.* (Utrecht, Netherlands, 2005), ACM New York, USA, 155-156.

[15] Min, W., and Wynter, L. 2011. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies, 19* (4), 606-616.

[16] Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A.L. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225* (1), 1-11.

[17] Psaraftis, H.N. 1995. Dynamic vehicle routing: Status and prospects. *annals of Operations Research, 61* (1), 143-164.

[18] Rizzoli, A.E., Oliverio, F., Montemanni, R., and Gambardella, L.M. 2004. Ant Colony Optimisation for vehicle routing problems: from theory to applications. *Galleria Rassegna Bimestrale Di Cultura, 9* (1), 1-50.

[19] Royce, W.W. Managing the development of large software systems. in *proceedings of IEEE WESCON*, (Los Angelos, USA, 1970), IEEE New York, USA, 1-9.

[20] Slater, A. 2002. Specification for a dynamic vehicle routing and scheduling system. *International Journal of Transport Management, 1* (1), 29-40.

[21] Solomon, M.M. 2000. VRPTW benchmark problems. Retrieved July 10, 2014, from: http://web.cba.neu.edu/∼msolomon/heuristi.htm.

[22] Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.Y. 1997 A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science, 31* (2), 170-186.

[23] Toth, P. and Vigo, D. *The vehicle routing problem.* SIAM, Philadelphia, USA, 2001.

[24] van Veen, B., Emmerich, M., Yang, Z., Bäck, T., and Kok, J. Ant Colony Algorithms for the Dynamic Vehicle Routing Problem with Time Windows. in *Natural and Artificial Computation in Engineering and Medical Applications,* (Mallorca, Spain, 2013), Springer-Verlag Berlin Heidelberg, 1-10.

[25] Weiss, G.M., and Hirsh, H. Learning to Predict Rare Events in Event Sequences. in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining.* (New York, USA, 1998), AAAI Press, Menlo Park, California, USA, 359-363.

Appendix: Driver form

Order nr	Postcode	Arrival time	Start time	Finish time	Waiting time (min)	Too late?	Stress (1-5)	Certain (1-5)	Comments (Stress? Why were you late? Any problems?)

Figure 8: The form which was given to the drivers during Trial 2.