BSc report TECHNISCHE WISKUNDE

"Models, Solutions and Relaxations of the Asymmetrical Capacitated Vehicle Routing Problem"

("Modellen, oplossingen en relaxaties van het asymmetrical capacitated vehicle routing problem")

LENNART KERCKHOFFS

Technische Universiteit Delft

Supervisor

Prof.dr.ir. K.I. Aardal

Committee Members

Dr. G.F. Nane

Dr. Ir. M. Keijzer

January, 2017

Delft

Preface

This bachelor thesis, or "Bachelor Eind Project (BEP)", is written as part for the curriculum of bachelor students of Applied Mathematics at the TU Delft. It is carried out at the TU Delft itself. Due to optimization being the most interesting and fun (to me) part of mathematics, I asked K. Aardal to be my supervisor. Together with her, we decided on the ACVRP. This decision was inspired by the growing popularity of ordering goods online, specifically online supermarkets.

This thesis should be read by any and all interested, but is mostly (in complexity) targeted at my fellow students in mathematics.

I would like to thank my commitee and Theresia van Essen, who helped me a lot when I had problems with AIMMS. I would like to extend a special thanks to my supervisor Karen Aardal, who greatly helped me through the process of coming up with the project, deciding the goal and determining ways to achieve this, not even mentioning the numerous times she read through this thesis to help me improve it.

Abstract

In this thesis, we take a look at the *Asymmetrical Capacitated Vehicle Routing Problem* (ACVRP). We will take a look at different possible formulations for the problem and choose one based on the ease of implementation, the computation speed of solving it, and the available relaxations. The problem, and its relaxations, will be modeled and solved using AIMMS, a commercial modeling software.
Using the methods described above, we model different cases and instances of the problem using a Two-Index Vehicle Flow formulation. We apply an Assignment Problem relaxation and a Linear Programming relaxation to each of the instances.
We find that the problem is easiest to solve when all customers are relatively close to each other (as opposed to being placed in separate clusters that are relatively far from each other), and that the LP relaxation gives bounds with a fairly good quality in short periods of time.

Contents

Chapter 1

Introduction

The optimization branch of mathematics concerns itself with a wide variety of problems (often inspired by practice). One could, for example, wonder what the fastest way is for their group of 13 friends with 4 vehicles available to pick everyone up from their home, and then meet at the bar.
That is in essence a comparable problem to what we will discuss in this thesis. We consider an online supermarket with a fleet of K vehicles with limited capacity. We want to use this fleet to serve a number of customers, each of which has a specific demand, while minimizing the total distance traveled by the fleet. We call this problem the Capaitated Vehicle Routing Problem (CVRP). To aid us in solving our problem to optimality, we will investigate some relaxations of the problem to get a lower bound on the optimal value, which is the shortest traveled distance in this case. The relaxations can also aid in determining the quality of solutions produced by heuristics, which provide upper bounds on the optimal value. Both the heuristics and the relaxations are computable in reasonable (polynomial) time. We will consider a specific case, inspired by the internet supermarket *Picnic*. This will be modeled with different formulations and then solved using the modelling software AIMMS. Finally, we will apply the determined relaxation using AIMMS.

1.1 Research goals

Vehicle routing problems are known to be very hard to solve, and thus it is unclear if online supermarkets can solve their instances to optimality in time to assign the routes to the customers. On top of that, the distances in our problem are not symmetric, meaning that traveling from client A to client B might be longer (or shorter) than going in the opposite direction. This gives rise to the so-called asymmetrical CVRP (ACVRP), which is at least as hard as the CVRP with symmetric distances. For this reason, they often approximate their solutions by using heuristic methods. While these generally give fairly accurate results, there is no way (or no time) for these online supermarkets to check the accuracy of their result. Using relaxations on some problem instances could produce some bounds on the solution to approximate the accuracy of a solution. The online supermarket we used as inspiration for our practical instances, called Picnic, has relatively small vehicles (meaning a relatively small capacity to store the groceries that need to be delivered). This means that delivery routes of the vehicles will be short. One goal of this thesis is to find out whether the short route ACVRPs are relatively easy to solve in practice.

On top of that, the delivery cars have two separate compartments for storing groceries: one for cooled, and one for uncooled goods. The customers' demands for groceries are also divided into a cooled and an uncooled demand. We want to adjust existing formulations for the ACVRP to account for the capacities and demands being split up.
We want to give a small overview of different formulations of the ACVRP, and find out which of the formulations is the most likely to yield the best results and also suitable for applying relaxations to.

1.2 Overview

In the coming chapters, we will slowly work toward achieving the goals outlined above. In Chapter 2 we will formally introduce the ACVRP, starting from the well-known Traveling Salesman Problem (TSP). After introducing the notation used, we will then consider some different formulations of the ACVRP. After that, in Chapter 3 we will define what relaxations are, and why they are useful. After that, we will consider different relaxations for some of the formulations of the ACVRP and their usefulness. In Chapter 4 we will shortly introduce AIMMS, the software used to model and solve our problem. After that, we will solve different instances of our problem, and apply relaxations to them. Finally, in Chapter 5 we will discuss our results.

Chapter 2

Problem Description

Now that we have our problem outlined, it is time to introduce it formally. First we will define a more specific case of the CVRP, the Traveling Salesman Problem. We do this because it is easier to understand than the CVRP, and extending it to the CVRP is convenient for explanation purposes. After that, we will consider different formulations for the ACVRP, and which to use when we start the implementation.

2.1 The Traveling Salesman Problem

The problem we are considering belongs to the family of Vehicle Routing Problems (VRPs). A VRP is a generalization of the well-known Traveling Salesman Problem (TSP). As an introduction to the VRP, and to make it easier to grasp, we will first introduce the specific VRP instance of the TSP.
The TSP is one of the most important and studied (graph theory-)optimization problems. Despite being formulated as early as 1930, it is still being studied extensively because of its complexity (in relation to the size of the problem) and its modern day relevance (from delivering mail to designing microchips).
In its simplest form, it can be represented as a mailman wanting to deliver his mail. He wants to do his job as quickly and efficiently as possible, and he starts from his own house and wants to end at his own house (or the post office). Thus, he tries to find the shortest route that starts and ends at his house, and visits all the other houses on his mail route as well. The name stems from a traveling salesman trying to sell his goods by visiting locations (often referred to as "cities") in the same way.
We model this problem by way of a graph G. We let every customer be a vertex in the vertex set V and connect them via arcs in the arc set A, representing the connections between the customers. Each arc is assigned an associated travel cost c_{ij}, usually the time or distance it takes to travel from one customer to another. We use a binary variable x_{ij} to check whether the arc traveling from i to j is used in the solution.
Below we give an Integer Linear Programming formulation that can be used to model and solve the TSP. Here, we assume we have $n-1$ houses where the mailman needs to deliver mail. House

1 is the mailmans house.

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1} c_{ij}^{n} x_{ij} \tag{2.1}$$

$$\text{subject to} \sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j = \{1, \ldots, n\} \tag{2.2}$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad i = \{1, \ldots, n\} \tag{2.3}$$

$$\text{subtour elimination constraints} \tag{2.4}$$

$$x_{ij} \in \{0, 1\} \tag{2.5}$$

Here, Constraints 2.2 and 2.3 are called the degree constraints, making sure that every customer is visited exactly once (Constraint 2.3 makes sure the mailman arrives once, and Constraint 2.2 ensures he also leaves each customer once). We will now explain what the Subtour Elimination Constraints (2.4) are.
A problem that sometimes arises while trying to solve a TSP is the concept of subtours. If we only include the "degree-constraints" (making sure every vertex is arrived at and left exactly once) in the problem formulation, small tours not starting or ending at the house of the mailman, called subtours, may occur. To this end, so-called "subtour elimination constraints" are added to the formulation to make it complete.

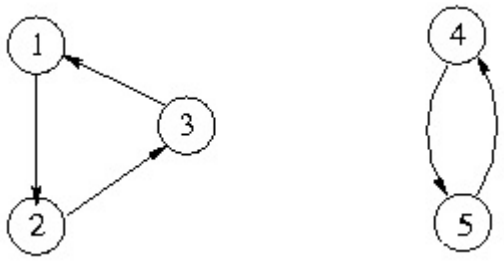

Figure 2.1: Subtours in a TSP with 5 customers

An example of these constraints are the Miller-Tucker-Zemlin constraints [12], which can also be found in the book by Papadimitriou [9]. Here, n is the number of customers and u_i, $i = 1, \ldots, n$ is an additional variable that gives an ordering to the vertices, excluding the depot, to prevent the formation of subtours:

$$u_i - u_j + n x_{ij} \leq n - 1, \; 1 \leq i \neq j \leq n \tag{2.6}$$

$$u_i \in \mathbb{R} \; \forall i = 1, \ldots, n \tag{2.7}$$

Constraints (2.6) force an ordering of the customers on a route. A proof that these inequalities not only restrict solutions to tours, but do not exclude any tours can be found in the book by Papadimitriou [9].
For more information on the TSP, we refer to the book by Applegate et al. [14]. Another set of constraints that excludes the appearance of subtours can be obtained using a nontrivial partition

$(S, \bar{S})$ of $\{0, \ldots, n\}$. We demand for each such partition that

$$\sum_{i \in S, j \in \bar{S}} x_{ij} \geq 1 \tag{2.8}$$

This makes sure that the sets in the partition are connected to each other. For example, in Figure 2.1, it would make sure that there is an arc going out of 1-2-3 into 4-5, and vice versa, as shown in Figure 2.2.

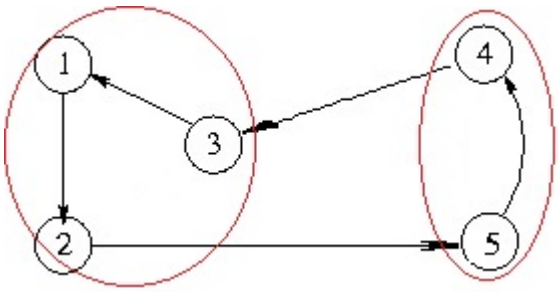

Figure 2.2: The partitions are now connected, no more subtours occur

Even though this formulation might seem exponentially large at first (because of the great number of subsets when the number of customers is high), it is actually the more used of the two. The problem is polynomially separable, meaning that after the problem is solved accounting only for the degree constraints, accounting for the subtour elimination constraints becomes a polynomially solvable Separation Problem. In such a problem, we must determine for a given vector x^* whether or not there exists a proper subset of vertices S such that $\sum_{i \in S, j \in \bar{S}} x_{ij} \geq 1$, meaning that a subtour elimination constraint is violated. See also the book by Applegate et al. [14]. For a more technical explanation, see the book by Grötschel et al. [15].
While the definition and aim of a TSP are not that complex to grasp, it can get computationally complicated rather quickly. An instance with n different houses/locations, for example, has $\frac{(n-1)!}{2}$ different solutions (that would be 1.814.400 different solutions for $n = 10$, and $1.216451 \cdot 10^{18}$ for $n = 20$). The TSP is NP-hard, which is Math for "very hard, we cannot consistently solve this in a short time (yet)" (NP stands for Non-deterministic Polynomial time problems).
Because of this, most TSPs are not solved exactly.

2.2 The ACVRP

The VRP can be derived from the TSP in the following way:
Instead of one mailman, we have a group of mailmen who all start from and end at the same post office (also called the *depot*). We want the total travel cost (defined in terms of travel time or distance usually) of all mailmen to be as small as possible.
If we then add that each house has a different amount of mail they will recieve (the *demand* of that home), and that each mailman has a bag with a limited space for mail and packages (a *capacity*), we get the *Capacitated* Vehicle Routing Problem (CVRP).
Now, if a mailman has to take different routes to and from one place to another (because of one-way streets, for example), the time travelled to and from one place to another will differ. In other words, the travel costs will be *asymmetrical*, and we obtain the asymmetrical Capacitated

Vehicle Routing Problem (ACVRP).
Because of the VRP being a generalization of the TSP, it is NP-hard as well.
We will mostly consider the ACVRP in this thesis, but other types of vehicle routing problems include:

- Generalized Vehicle Routing Problem (GVRP): we group several houses (or maybe a neighbourhood) into a so-called cluster. Then, we say that only one house from a cluster has to be visited.
- Vehicle Routing Problem with Pickup and Delivery (VRPPD): some mail needs to be moved from one location to the other, so the mailmen have some pick-up location they need to visit before they deliver their mail.
- Vehicle Routing Problem with LIFO: the VRPPD, with an added Last-In-First-Out constraint: the package being delivered must be the most recently picked up one. Because items do not need to be temporarily unloaded to be able to deliver items further in the back (or: mailmen do not have to look through their bag, as the package is on top), reducing delivery times.
- Vehicle Routing Problem with Multiple Trips (VRPMT): each mailman is able to do more than one route.
- Open Vehicle Routing Problem: mailmen do not have to return to their house/the depot.
- Vehicle Routing Problem with Time Windows (VRPTW): people need to be home to accept their packages, and there are only certain times each day where they are able to do so. We add to each customer a drop off time and make sure packages are delivered after each customers earliest-, and before his/her latest arrival time.

These are, of course not all of the possible variations. For more examples, one could for example take a look at the overview presented by Escobar et al. [3].
In our specific case, instead of houses recieving mail, we have customers ordering groceries from an online supermarket. We replace the mailmen with vehicles and the mail with groceries. The post office we now just call a depot.

In the case of the online supermarket Picnic, a fleet of K vehicles needs to deliver groceries to customers with a certain demand for groceries. Each customer has a given demand for cooled and uncooled goods. Analoguously, the vehicles, depicted in Figure 2.3, in the fleet have two capacities: one for cooled, and one for uncooled goods. We assume that these capacities are the same. We define the travel cost from a vertex to another as the distance between these two vertices. These distances are not symmetric because of one-way roads. Thus, we obtain our ACVRP (with two capacities).
Picnic promises their customers to give them an accurate estimate of the arrival of their grocceries (a 5 minute time window). However, this does not add time windows to our ACVRP, as customers are first assigned to a certain shift or time window, and then the routes for the vehicle are determined. Thus, the time window part of this problem is a separate problem that happens before solving the ACVRP. However, this does imply short routes. All customers need to be able to be served within an hour, so, considering drop-off times and possible delays, we want routes of at most 45 minutes. These "routes" start at the first, and end at the last customer, they do not count the time traveling to or from the depot. We do this because the vehicles can in theory

depart from the depot an arbitrary amount of time before the groceries need to be delivered at the first customer. They can then wait with visiting all the customers until the time the groeries are promised to be delivered to the first customer in their route.
For the sake of argument, we assume that a vehicle goes 30 km/h (which is equal to 8.333 m/s) on average during their delivery route (not accounting for time standing still when dropping off goods), and they have a drop-off time of 4 minutes for every customer. This could prompt us to add an extra constraint

$$\frac{\sum_{(i,j)\in V_{r_i}} c_{ij}x_{ij}}{8.333} \cdot 60 + 4 \cdot n_{r_i} \leq 45 \tag{2.9}$$

See Section 2.3 below for a description of the variables and coefficients used.
Thus, the maximum length of the route is equal to

$$\frac{45 - 4 \cdot n_{r_i}}{60} \cdot 8.333 \text{ meters} \tag{2.10}$$

An important question to consider is whether an ACVRP with short routes is computationally easier to solve in practice than an ACVRP with routes of arbitrary length.
If we test this out using AIMMS, this turns out to be the case: for a network with 10 customers and distances between customers short relative to the distance between the depot and customers, we consider two cases:

- one with a fleet of 8 vehicles with capacity 13, constituting short routes
- one with a fleet of 3 vehicles with capacity 26, constituting longer routes

AIMMS was able to solve the problem with short routes in around 0.06 seconds, and the problem with longer routes in 0.38 seconds. While 0.38 seconds is of course still a very fast time to solve the problem, this does illustrate that problems with long routes in the solution take longer to solve than problems with short routes. In bigger problems with for example up to 100 customers, these small computation differences (for small instances) can become quite significant.
This is probably because the amount of customers to be considered to use in a route decreases with each route that was previously computed. So with short routes, each route becomes increasingly more easy to compute, faster than with longer routes where this "elimination" of customers does not happen as often.

Figure 2.3: E-workers: Picnics delivery cars

2.3 Formal definition and notation

Let V be a vertex set $\{v_0, v_1, \ldots, v_n\}$, with v_0 representing a depot, and the other v_i, $i = 1, \ldots, n$ representing customers.
Let A be a set of arcs, $A = \{(v_i, v_j) | \, \forall i \neq j\}$ with pairs of customers representing a road between these customers. We want our graph to be complete, so if there is no direct road between two customers, we add an arc between them with cost equal to the shortest path between them. Another option on could present would be to add an arc with a very high cost to make the graph complete, but this would exclude certain feasible solutions of the problem).
Let x_{ij} be a binary variable.

$$x_{ij} = \begin{cases} 1 \text{ if arc } (i,j) \text{ is traversed in the solution to the ACVRP} \\ 0 \text{ otherwise} \end{cases}$$

Let $C = \{c_{ij} | i, j \in \{0, \ldots, n\}\}$ be a set of costs associated with traveling from v_i to v_j (note that $c_{ij} = c_{ji}$ does not always hold because our problem is asymmetrical).
Each customer i has two demands d_{i_1} and d_{i_2} (or one demand d_i in some of our examples) associated with the amount of uncooled and cooled items they ordered. Note that $d_{0_1} = d_{0_2} = 0$. In this case, the demands are integer, because goods are allocated into crates of equal size (also called totes), and each storage of a vehicle (called "E-workers" in Picnic's case, see Figure 2.3). has a given number of spaces for these crates.
We also have a fleet of K vehicles with two associated capacities Q_1, Q_2 (one capacity Q in some of our examples) for uncooled and cooled storages (we assume a homogeneous fleet here, and we assume that $Q_1 = Q_2$).
For the solution, we assign each customer route $r_i = (v_0, v_{r_{i_1}}, v_{r_{i_2}}, \ldots, v_0)$ to a vehicle in the fleet. The total demands of all these customers can of course not exceed the capacities of the vehicle, the route has to conform with the time windows, and customer orders cannot be divided over different vehicles.
$V_{r_i} = \{v_0, v_{r_{i_1}}, \ldots\}$ is the set of customers in a route r_i, so $|V_{r_i}| = n_{r_i}$ is the number of customers in a given route. We denote $\mathbf{R}$ as the set of all customer routes.
Finally, $\gamma(S), S \subset V$ is the minimum number of vehicles needed to serve all customers in a set S. $\gamma(S)$ can be determined by solving a Bin Packing Problem (BPP), where you have n containers with a certain capacity, and m units of different size that all need to be assigned to a certain container.

2.3.1 Solution

A solution to the problem consists of a collection of feasible routes $R = \{r_1, \ldots, r_K\}$ such that each customer is visited exactly once. Note that in cases where the size of the fleet is not determined beforehand, R can contain empty routes.
In this thesis, we define the 'cost' of a route to be the sum of the travel distances. Other examples of cost include travel time, fuel costs or combinations of any of the previous three.

2.4 Different formulations

In the literature, there are more or less two basic modelling approaches for the VRP. The first type are called the *vehicle flow formulations*. In a flow problem, a certain amount of flow needs to be transported from the source node to the sink node. Each arc has a certain capacity for the flow associated with it.
These formulations use integer variables associated with each arc of the graph, indicating whether an arc is traversed or not. However, the linear programming (LP) relaxations of these models can be very weak when combined with tight capacity constraints. The ILP formulations can usually be modeled as either a two-index or three-index problem. In a two-index problem, we use a variable x_{ij} to indicate whether the arc (i, j) is traveled in the solution of the problem. In three-index formulations, we are a little more specific and use a variable x_{ijk} to indicate whether vehicle k travels from i to j in the solution of the problem. We will only consider two-index formulations in this paper, not three-index formulations. We do this because in our case, when the vehicles are uniform, two-index formulations achieve the same results as three-index formulations, but they are usually a little faster. The paper by Baldacci et al. [1] shows that the quality of both formulations is the same, but that two-index formulations are computationally less demanding.

The other type of formulation has an exponential number of binary variables, each associated with a different feasible circuit. We try to determine a collection of circuits with minimum cost, such that each customer is served once. We call this a *set partitioning problem* (SPP). These kinds of formulations are usually solved with a method called column generation. More information and specifics on this method can be found in the book by Wolsey [10]. Unfortunately, due to time constraints and the scope of this thesis, we will not consider column generation any further. Because of this, and the large amount of variables in SPP formulations, we will mainly consider the flow formulations in this thesis.

2.4.1 Two-Index Vehicle Flow

The formulation given below is based on the formulation for the ACVRP in the paper by Toth and Vigo [2].

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.11}$$

$$\text{subject to } \sum_{i \in V} x_{ij} = 1 \text{ for all } j \in V \setminus \{0\} \tag{2.12}$$

$$\sum_{j \in V} x_{ij} = 1 \text{ for all } i \in V \setminus \{0\} \tag{2.13}$$

$$\sum_{i \in V} x_{i0} = K \tag{2.14}$$

$$\sum_{j \in V} x_{0j} = K \tag{2.15}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \gamma(S) \text{ for all } S \subset V \setminus \{0\}, S \neq \emptyset \tag{2.16}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j \in V \tag{2.17}$$

In this formulation we can clearly see the similarity with the TSP formulation from Section 2.1. Here, Constraints (2.12) and (2.13) make sure that each customer is visited (arrived at and left from) exactly once. Constraints (2.14) and (2.15) make sure all vehicles end and start at the depot. Finally, Constraint (2.16) ensures that for each subset of customers, there are enough vehicles to serve all of them (thus eliminating the possibility of subtours appearing in a solution). The problem with Constraint (2.16) is that there is a very large number of subsets S of V. For a problem with 10 customers, there are 10468900 different subsets S, and we cannot use the same method of separation as with the TSP. This is most likely because the multitude of vehicles making the problem not polynomial anymore.
Thus, we want to find a constraint that is somewhat easier on the calculations. Toth and Vigo [2] proposed considering the subtour elimination constraints for the TSP by Miller et al. (1960), and extending them to the CVRP. These new constraints replace Constraint (2.16) with the following two constraints:

$$u_i - u_j + Q x_{ij} \leq Q - d_j \text{ for all } i, j \in V \setminus \{0\} \tag{2.18}$$

$$d_i \leq u_i \leq Q \tag{2.19}$$

where $u_i, i \in V \setminus \{0\}$ is a continuous variable, analoguous to the one used in the subtour elimination constraint of the TSP. Constraints (2.18) and (2.19) impose the capacity requirements of the CVRP. Also, when $x_{ij} = 0$, Constraint (2.18) is not binding since $u_j \leq C$, and $u_i \geq d_i$ (else customer i cannot be serviced). When $x_{ij} = 1$, it imposes that $u_i \geq u_j + d_i$ (note that when $x_{ij} = 1$ we first arrive at i, and then go to j straight after).
Toth and Vigo [2] proposed an extra requirement, $d_i + d_j \leq Q$, for Constraint (2.18), however the capacity is already implied by the combination of Constraints (2.18) and (2.19) for both $x_{ij} = 1$ and $x_{ij} = 0$, and accounting for these extra requirements allows for certain infeasible solutions where the total load of a route exceeds the demand (especially in short routes).

2.4.2 Two-commodity flow Formulation

A multi-commodity flow problem is a flow problem where there are multiple flow demands between the source and sink nodes.
The two-commodity flow forulation of the CVRP is based on the two-commodity flow formulation introduced by Finke et al. [7] for the TSP.
In this case, $V = \{0, \ldots, n+1\}$, where $0, n+1$ represent the same vertex: the depot. We define y_{ij} as follows:

$$y_{ij} = \begin{cases} \text{the amount of groceries carried from customer } i \text{ to customer } j, \text{ if } x_{ij} = 1 \\ \text{the amount of space left in the vehicle, } y_{ji} = Q - y_{ij} \text{ if } x_{ji} = 1 \\ 0 \text{ otherwise.} \end{cases}$$

We define $d(S)$ for $S \subset V$ be the total demand from the customers in S. $\delta(S)$ is the set of customers incident to (a customer in) the set S. $\bar{A}$ is the set of arcs, not considering the arcs to and from vertex $n+1$. The two-commodity flow formulation for the ACVRP, based on the formulation by Baldacci et al. [1] then becomes:

$$\min \sum_{\{i,j\} \in \bar{A}} c_{ij} x_{ij} \tag{2.20}$$

$$\text{subject to} \sum_{j \in V \setminus \{n+1\}} (y_{ji} - y_{ij}) = 2d_i, \forall i \in V \tag{2.21}$$

$$\sum_{j \in V} y_{0j} = d(V) \tag{2.22}$$

$$\sum_{j \in V} y_{j0} = KQ - d(V) \tag{2.23}$$

$$\sum_{j \in V} y_{n+1,j} = KQ \tag{2.24}$$

$$\sum_{\{i,j\} \in \delta(\{h\})} x_{ij} = 2, (\forall h \in V) \tag{2.25}$$

$$y_{ij} + y_{ji} = Q x_{ij}, (\forall \{i,j\} \in A) \tag{2.26}$$

$$y_{ij} \geq 0, y_{ji} \geq 0, (\forall \{i,j\} \in A) \tag{2.27}$$

$$x_{ij} \in \{0,1\}, (\forall \{i,j\} \in A) \tag{2.28}$$

Here, Constraint (2.21) makes sure each customer gets exactly their demand delivered to them: i is visited exactly once, i.e. there is exactly one car visiting i from exactly one customer, p (which can also be the depot in this case), and that car departs to exactly one other customer q (not the depot). From this, it follows that $\sum_{j \in V \setminus \{n+1\}} (y_{ji} - y_{ij}) = y_{pi} - y_{ip} + y_{qi} - y_{iq}$. Let d_p, d_i be the demands of i and p respectively, and $\bar{d}_q$ the total demand of q and all other customers on this particular route that are visited after q.
Then, $y_{pi} - y_{ip} + y_{qi} - y_{iq} = d_i + \bar{d}_q - (Q - (d_i + \bar{d}_q)) + (Q - \bar{d}_q) - \bar{d}_q = d_i + d_i + \bar{d}_q + \bar{d}_q - \bar{d}_q - \bar{d}_q - Q + Q = 2d_i$.
Constraints (2.22) and (2.23) make sure all the flow (groceries) leave from the depot (the flow carried away from the depot is equal to the total demand of all the customers), and that as much flow as the total demand is dropped off (the total flow coming into the depot is equal to the total capacity of the vehicles, minus the total demand of all of the customers). Constraint (2.24)

ensures all vehicles end up back at the depot again. Constraint (2.25) forces any feasible solution to contain two edges incident to each other, while Constraint (2.26) ensures that the sum of the load in the vehicle plus the amount of space left in the vehicle is equal to the capacity of the vehicle.
The two-commodity formulation can be rewritten in terms of variables y_{ij} only. To do this, we substitute variables x_{ij} with variables y_{ij} using Constraint (2.26). The Constraints (2.28) should then be replaced by $y_{ij} + y_{ji} \in \{0, Q\}, \forall\{i, j\} \in \bar{A}$.
All valid inequalities that can be used to relax the ACVRP can be applied to the two-commodity flow formulation by changing the x_{ij} to an y_{ij} in the formulation.
The relaxations for the two-commodity flow formulation are of a quality equal to that of the two-index vehicle flow formulation. However, the LP relaxation of the two-commodity flow is easier and faster to solve than the LP relaxation of the two-index vehicle flow formulation (see also the paper by Baldacci et al. [1]).
Despite this, the two-index vehicle flow formulation has been studied more and therefore, more and better relaxations are known.

2.4.3 Set partitioning formulation

The set partitioning formulation of the CVRP was originally proposed by Balinski and Quandt [8].
This formulation is different from the ones before in that we do not try to find paths between customers, which then form different routes. Instead, we consider the set of all feasible routes $\mathbf{R}$, and try to find which of these routes minimize the total travel costs.
Here, a_{jr_i} is a binary coefficient that has value 1 if customer j belongs to route r_i, and 0 otherwise. ξ_{r_i} is a binary variable that is equal to 1 if and only if r_i belongs to the optimal solution.
[8].

$$\min \sum_{r_i \in \mathbf{R}} c_{r_i} \xi_{r_i} \tag{2.29}$$

$$\text{subject to } \sum_{r_i \in \mathbf{R}} a_{jr_i} \xi_{r_i} = 1 \ (\forall j \in V \setminus \{0\}) \tag{2.30}$$

$$\sum_{r_i \in \mathbf{R}} \xi_{r_i} = K \tag{2.31}$$

$$\xi_{r_i} \in \{0, 1\} \ (\forall r_i \in \mathbf{R}) \tag{2.32}$$

Constraint (2.30) makes sure that each customer is covered by one route, and Constraint (2.31) requires that K routes are selected.
When the cost matrix satisfies the triangle inequality [1], we can obtain a so-called *set covering formulation* that preserves the optimal objective function value by writing Constraint (2.30) as $\geq$ inequalities.
This formulation is very general and can take additional constraints such as time windows or route lengths into consideration, since route feasibility is implicitly considered in the definition of the index set of all feasible routes $\mathbf{R}$.
Another advantage of the set partitioning formulation is that any valid inequality designed for the two-index formulation can be transformed into a valid inequality for the set partitioning

[1] In this case, traveling over a direct path from A to B will always be less expensive than traveling from A, to C and then to B.

formulation by noting that a solution ξ for the set partitioning formulation can be transformed into a solution x of the two-index formulation, using:

$$x_{ij} = \sum_{l \in \mathbb{R}} \eta_{ij}^{l} \xi_l, \, \forall \{i,j\} \in E \tag{2.33}$$

where the η_{ij}^{l} are defined as follows:

- if l is a single customer route covering customer h, then $\eta_{0h}^{l} = 2$ and $\eta_{ij}^{l} = 0$, $\forall \{i,j\} \in E \setminus \{0,h\}$
- if l is not a single customer route, then $\eta_{ij}^{l} = 1$ for each edge $\{i,j\}$ covered by route ξ and $\eta_{ij}^{l} = 0$ otherwise.

Then, any valid inequality designed for the two-index formulations can be transformed into a valid set partitioning formulation inequality.
More information on the set partitioning formulation and valid bounds for this formulation can be found in the paper by Baldacci et al. [1].
However, as mentioned above, we will not consider this formulation any further because of the scope of this thesis.

Chapter 3

Relaxations

Sometimes a problem is too large or complicated to solve exactly (within a reasonable time window). In such a case, other methods must be implemented to find a near-optimal solution, or to approximate one. Some options in the first case include applying heuristic methods, using algorithms that find a solution that conforms to the constraints, but not necessarily giving the best objective value.
Another way is to approximate the problem and solving a nearby, related problem. This related problem can for example be achieved by removing or weakening some constraints of the problem. The solution to the related problem provides a bound on the objective value for the problem: a lower bound for a minimization problem, and an upper bound for a maximization problem.
Suppose we solve a minimization problem using heuristics. This yields an upper bound on the optimal value of the solution. To then assess the accuracy of this upper bound, we could apply a relaxation to the problem, producing a lower bound on the optimal value of the solution. Comparing these two bounds should give a decent indication of the quality of solutions, for example those found with heuristic methods.
Formally, a relaxation of a minimization problem $z = \min\{c(x) : x \in X \subseteq \mathbb{R}^n\}$ is another minimization problem $z = \min\{c'(x) : x \in T \subseteq \mathbb{R}^n\}$, such that

- $X \subseteq T$
- $c'(x) \leq c(x)\ \forall x \in X$

For further information, see the book by Wolsey [10].
One could wonder if there is a way to use the bounds from heuristics and relaxations, and to narrow them down to achieve the optimal solution.
The most well-known way of doing this is using branch-and-bound. With this, the solution space is partitioned into convex sets, for which upper and lower bounds are computed. In a branch and bound algorithm, the set of all possible solutions is iteratively split into smaller sets. The cost function is then minimized over these smaller sets. We call this *branching*, as this can be seen as roots of a tree that are splitting up, or branching out.
Because we do not want to check every possible smaller set, we also introduce the *bounding* procedure: the algorithm keeps tracks of the bounds on the minimal cost function that it is looking for, and uses these bounds to "prune" the solution space, meaning that it eliminates sets of possible solutions that will not contain a useful minimum on the value of the cost function.

While branch-and-bound is a nonheurstic method that returns an optimal solution to the problem, it is sometimes very slow (in the worst case even exponentially so), and a bit of luck is needed to use it to get a solution within a reasonable time window. Having good bounds also typically helps to speed up the process.
We give three examples of relaxations: LP relaxations, Lagrangian relaxations and problem-specific relaxations.

3.1 LP relaxations

In the case of an 0-1 Integer Linear Program (ILP), we obtain the LP-relaxation by removing the integer restriction on the variables x_{ij}, allowing them to be continuous and part of the interval [0,1]. The good thing about LP-relaxations is that it derives from the NP-hard ILP problem a related Linear Program, which typically is much easier to solve, as LP problems are polynomially solvable.

3.2 Lagrangian relaxations

Lagrangian relaxations are very useful when the problem has constraints that can be divided into two "categories": "good" constraints, which are fairly easily to solve the problem with, and "bad" constraints, the constraints that make the problem hard to solve. The problem is then "relaxed" by removing the bad constraints, making the problem easier to solve. To still somewhat account for these contraints, they are added into the objective function with certain weights. These weights are called the Lagrangian Multipliers.
Another, more detailed explanation, can be found in the book by Wolsey [10]. Other examples, and a short explanation, of Lagrangian Relaxations can be found in the paper by Klau [4].

3.3 Problem-specific relaxations

Often, a problem can be relaxed in a way unique to that problem. These problem-specific relaxations include the well-known 1-tree relaxation for the TSP. A 1-tree is a graph with vertices $1, 2, \ldots, n$ consisting of a tree on $2, 3, \ldots n$ and two arcs incident with vertex 1. It is a tree with exactly one cycle, which contains vertex 1. This relaxation was proposed by Held and Karp [6]. A one-tree relaxation is applied as follows: choose a node in de graph, for example node i. We then try to find the cheapest one-tree on the graph, such that the cycle in the one-tree includes i, and for which i is incident to two other nodes. Note that a cycle in a 1-tree that includes all the vertices in the graph is a valid tour and as such a feasible solution for the TSP. Thus, finding the aforementioned cheapest one-tree including i is a relaxation, as the optimal solution is a 1-tree including i, but it does not have to be the cheapest 1-tree including i.
Clearly this problem is not an LP-relaxation, as we do not change any restraints on the variables x_{ij}. It is not a Lagrangian relaxation either, as we do not change the objective function in any way.

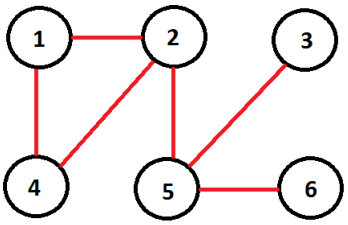

Figure 3.1: An example of a 1-tree on a graph with 6 nodes

3.4 Relaxations for the two-index vehicle flow formulation

We will now consider some possible problem-specific relaxations for the two-index vehicle flow formulation and compare their performance, before applying one later in Chapter 4.
The following relaxations are based on the formulation for the ACVRP in the paper by Toth and Vigo [2].

3.4.1 The LP-relaxation

As described in Section 3.1. The formulation for the LP-relaxed two-index vehicle flow formulation is:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3.1}$$

$$\text{subject to } \sum_{i \in V} x_{ij} = 1 \text{ for all } j \in V \setminus \{0\} \tag{3.2}$$

$$\sum_{j \in V} x_{ij} = 1 \text{ for all } i \in V \setminus \{0\} \tag{3.3}$$

$$\sum_{i \in V} x_{i0} = K \tag{3.4}$$

$$\sum_{j \in V} x_{0j} = K \tag{3.5}$$

$$u_j - u_i + Q x_{ij} \leq Q - d_i \text{ for all } i, j \in V \setminus \{0\} \tag{3.6}$$

$$d_i \leq u_i \leq Q \tag{3.7}$$

$$x_{ij} \in [0, 1] \text{ for all } i, j \in V \tag{3.8}$$

3.4.2 The assignment lower bound

In this relaxation, we will ignore Constraints (2.16) (or (2.18) and (2.19)). The problem we then obtain is an asymmetrical K-TSP problem (a regular TSP, only with K mailmen instead of one),

without the subtour elimination constraints:

$$\min \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} \tag{3.9}$$

$$\text{subject to } \sum_{i\in V} x_{ij} = 1 \text{ for all } j \in V \setminus \{0\} \tag{3.10}$$

$$\sum_{j\in V} x_{ij} = 1 \text{ for all } i \in V \setminus \{0\} \tag{3.11}$$

$$\sum_{i\in V} x_{i0} = K \tag{3.12}$$

$$\sum_{j\in V} x_{0j} = K \tag{3.13}$$

$$x_{ij} \in \{0,1\} \text{ for all } i,j \in V \tag{3.14}$$

In this problem, we want a collection of circuits of our graph that visits all vertices (not 0) once, and vertex 0 K times, while minimizing the total distance traveled. Because we do not account for the capacity and disregard subtour elimination constraints, it is possible for this solution to be infeasible for the ACVRP.
This problem is a little easier to solve when we transform the problem above into an *assignment problem* (AP). An AP is a case of a *transportation problem* (TP). In a transportation problem, we want to find an optimal transportation scheme on a graph with m warehouses and n retail outlets, while keeping in mind the demands of each customer and the supply of each warehouse. The goal in an AP is to minimize the cost or time of completing a number of jobs by some number of people. Important to note here is that the number of sources is equal to the number of destinations (because each person is assigned, and assigned to exactly one job). If the number of people is different from the number of jobs, we can compensate by adding "dummy" jobs or people, until the number of sources is equal to the number of sinks.
We define this problem on an extended complete digraph $G' = (V', A')$, where $V' := V \cup W'$ and $W' = \{n+1, \ldots, n+k-1\}$ contains $K-1$ additional copies of vertex 0, and the cost c'_{ij} of each arc in A' is defined as follows:

$$c'_{ij} = \begin{cases} c_{ij} \text{ for } i,j \in V \setminus \{0\} \\ c_{i0} \text{ for } i \in V \setminus \{0\}, j \in W' \\ c_{0j} \text{ for } i \in W', j \in V \setminus \{0\} \\ \lambda \text{ for } i,j \in W' \end{cases} \tag{3.15}$$

where λ is a very large number M (see also Figure 3.2).
Now, we can replace (2.14) by K constraints of type of (2.12), one for each copy of the depot. In the same way, we can replace (2.15) with K constraints of type of (2.13).
Originally, this transformation was used by Lenstra and Rinnooy Kan (1975) to transform a TSP into a m-TSP (where we want m circuits visiting a certain vertex m times, and all other vertices once).
Note that we arrive at a different transformation if we define λ differently. For example, $\lambda = 0$ means we try to find a solution with at most K routes, and $\lambda = -M$ means we try to find a solution with K_{min} routes.

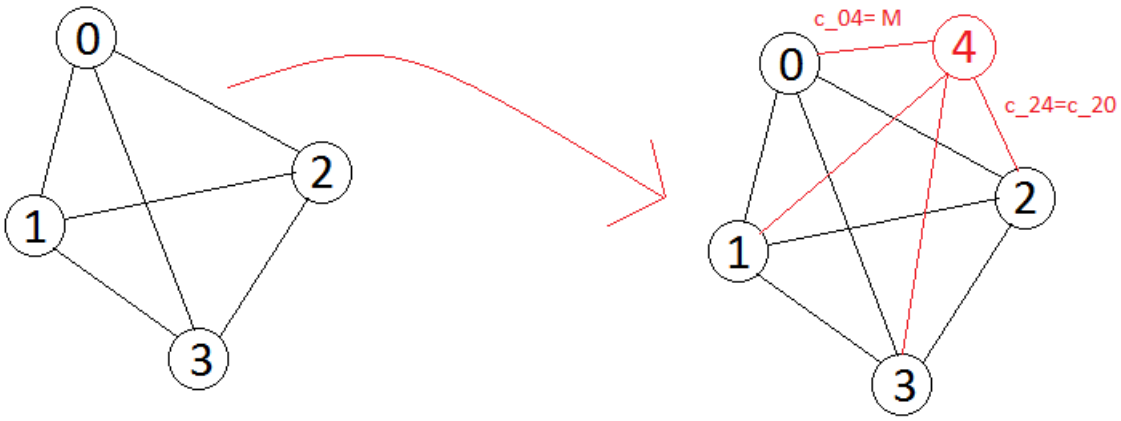

Figure 3.2: The extended complete digraph G' on a network with 2 vehicles, giving us one copy of the depot, 0, to add to the graph.

3.4.3 K-Shortest spanning Arborescence problem (KSSA)

This relaxation is based on the 1-tree relaxation for the symmetric TSP by Held and Karp [6]. We convert the flow problem into an arborescence: a directed graph with a root u, where for every other vertex v there is exactly one directed path from u to v.
We obtain this problem from the two-index vehicle flow formulation by removing the outdegree Constraints (2.12) for all customer vertices ($i = 1, \ldots, n$), and weakening the capacity-cut Constraints (2.16) so only the connectivity is important (by replacing the right-hand side with 1).
We then obtain the *K-shortest spanning arborescence problem (KSSA)*:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3.16}$$

$$\sum_{i \in V} x_{ij} = 1 \text{ for all } j \in V \setminus \{0\} \tag{3.17}$$

$$\sum_{i \in V} x_{i0} = K \tag{3.18}$$

$$\sum_{ij \in V} x_{0j} = K \tag{3.19}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq 1 \text{ for all } S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{3.20}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j \in V \tag{3.21}$$

This problem can be solved by considering two separate subproblems:

- determining a min-cost spanning arborescence with outdegree K at the depot vertex, defined by (3.16), (3.17), (3.20) and (3.21)
- determining a set of K min-cost arcs entering the depot, defined by (3.16), (3.18), and (3.19).

The first subproblem can be solved in $O(n^2)$ time, and the second problem in $O(n)$ time. Thus, this relaxation can be solved in $O(n^2)$ time.

This relaxation is seldomly used in branch and bound algorithms, and its quality is generally poor (or at least worse than the AP lower bound).

3.4.4 The disjunctive lower bound

We will consider two bounds here, both proposed by Fischetti et al. [13]. One bound is based on a disjunction on infeasible arc subsets, and the second bound is based on a min-cost flow relaxation.
We will first define what an infeasible arc subset really is.
A given arc subset $B \subset A$ is infeasible if no feasible solution to the ACVRP can use all its arcs. More formally, when $\sum_{(i,j)\in B} x_{ij} \leq |B| - 1$ is valid for the ACVRP.
For any infeasible arc subset $B \subset A$, the following logical disjunction holds for all feasible solutions x of the ACVRP: $\bigvee_{(i,j)\in B}(x \in Q^{ij} := \{x \in \mathbb{R}^A : x_{ij} = 0\})$ ($\mathbb{R}^A = \mathbb{R}^k$, where $|A| = k$). In other words, for any infeasible arc subset B, at least one of the arcs contained in B is used in none of the feasible solutions x of the ACVRP.
Then, we have defined $|B|$ restricted problems, each denoted as RP^{ij}, including the additional condition $x_{ij} = 0$ imposed for a different $(i,j) \in B$. For each RP^{ij}, a lower bound z^{ij} is computed through the AP relaxation (with $c_{ij} := +\infty$ to impose $x_{ij} = 0$). Since $z^{ij} \geq L_{AP}$ for all $(i,j) \in B$, the disjunctive bound $L_D :=\min\{z^{ij} : (i,j) \in B\}$, is of better quality than L_{AP}.
We will now describe a way to acquire infeasible subsets B:
First we solve the AP relaxation with no additional constraints, and store the corresponding solution $(x^*_{ij} : i,j \in V)$. When x^* is feasible for the ACVRP, the lower bound L_{AP} cannot be improved. Because then, the relaxation yields a feasible problem and we have solved the problem itself. Otherwise, try to improve it by using a disjunction on a suitable infeasible arc subset B. Note that imposing $x_{ij} = 0$ for any $(i,j) \in A$ such that $x^*_{ab} = 0$ would produce $z^{ij} = L_{AP}$. Therefore, B is chosen as a subset of $A^* := \{(i,j) \in A : x^*_{ij} = 1\}$, such that it corresponds to one of the following:

- a circuit which is disconnected from the depot vertex
- a path such that the total demand of the associated customer vertices exceeds Q
- a feasible circuit which does not visit a set of customers, S, with a total demand that cannot be served by the remaining $K - 1$ vehicles ($\gamma(S) > K - 1$)

if any exist. A more precise description of an algorithm that can be used to acquire the disjunctive lower bound can be found in the paper by Fischetti et al. [13].
Regrettably, due to time constraints, we are not able to implement this bound in this thesis.

3.4.5 The Lower bound based on flow

Consider a partition of V: $\{S_1, \ldots, S_m\}$ with $0 \in S_1$. Now, partition A into $\{A_1, A_2\}$, where A_1 contains the arcs with both end points in S_h, $h = 1 \ldots m$, and A_2 those connecting vertices belonging to different S_h's:

$$A_1 := \bigcup_{h=1}^{m} \{(i,j) \in A : i,j \in S_h\} \tag{3.22}$$

$$A_2 := A \setminus A_1 \tag{3.23}$$

Different vertex partitions $\{S_1, \ldots, S_m\}$ lead to different lower bounds. Choosing $S_h = \{h\}$, for example, produces a relaxation that coincides with the AP relaxation. When S_hs with a cardinality greater than one are chosen, the relaxation can take into account the associated capacity-cut constraints (which are neglected by the AP relaxation), while making a possible contribution of the arcs in S_h (belonging to A_1) less strong and weakening the degree constraints of the vertices in S_h.
We will now decribe a bound based on projections, L_P, given by $L_P := z_1 + z_2$, where the z_i are lower bounds on $\sum(c_{ij} : (i,j) \in A^* \cap A_t)$ for every optimal ACVRP solution $A^* \subset A$.
Initially, we ignore the contribution of the arcs internal to S_h to L_P, meaning that we set z_1 equal to 0 (this will be clarified below).
We compute z_2 by solving the LP relaxation R1 obtained from the two-index vehicle flow formulation by weakening Constraints (2.12)-(2.15) to account for the removal of the arcs in A_1, and imposing the Constraints (2.16) only for the m subsets S_h.

The LP relaxation model is:

$$z_2 = \min \sum c_{ij} x_{ij} \tag{3.24}$$

$$\text{subject to} \sum_{i \in V:(i,j) \in A_2} x_{ij} \leq \begin{cases} 1 \text{ for all } j \in V \setminus \{0\}; \\ K \text{ for } j = 0; \end{cases} \tag{3.25}$$

$$\sum_{j \in V:(i,j) \in A_2} x_{ij} \leq \begin{cases} 1 \text{ for all } i \in V \setminus \{0\}; \\ K \text{ for } i = 0; \end{cases} \tag{3.26}$$

$$\sum_{i \notin S_h} \sum_{j \in S_h} x_{ij} = \sum_{i \in S_h} \sum_{j \notin S_h} x_{ij} \geq \begin{cases} \gamma(V \setminus S_h) \text{ for } h = 1; \\ \gamma(S_h) \text{ for } h = 2, \ldots, m; \end{cases} \tag{3.27}$$

$$x_{ij} \in \{0, 1\} \text{ for all } (i,j) \in A_2 \tag{3.28}$$

We can view this problem as an instance of a min-cost flow problem on an auxiliary layered network, making it easier to be solved efficiently. The network contains $2(n + m + 2)$ vertices:

- two vertices, i^+ and i^-, for all $i \in V$
- two vertices, a_h and b_h for all $h = 1, \ldots, m$
- a source vertex s and a sink vertex t

with arcs and costs:

- for all $(i,j) \in A_2$: arc (i^+, j^-) with cost c_{ij} and capacity 1
- for all $h = 1, \ldots, m$: arcs (a_h, i^+) and (i^-, b_h) for all $i \in S_h$, with cost 0 and capacity 1 for $i \neq 0$ and K for i =0
- for all $h = 1, \ldots, m$: arc $(a_h, b_h$ with cost 0 and capacity $|S_h| - \gamma(S_h)$ if $h \neq 1$ or $|S_1| + K - 1 - \gamma(V \setminus S_1)$ if $h = 1$
- for all $h = 1, \ldots, m$: arcs (s, a_h) and (b_h, t), both with cost 0 and capacity $|S_h|$ if $h \neq 1$ or $|S_1| + K - 1$ if $h = 1$

Finding the min-cost $s - t$ flow of value $n + K$ on the above network solves the LP relaxation model. The worst-case time complexity for computing z_2 and the corresponding residual costs is $O(n^3)$.
Due to the scope of this project, we will not consider this bound any further though.

3.5 Relaxations for the two-commodity flow formulation

An overview of different bounds on the commodity flow (and on another two-index flow) formulation of the ACVRP can be found in the paper by Baldacci et al. [1].
The LP lower bound of the two-index formulation with rounded capacity (RC) inequalities (a weaker version of Constraint (2.16)) replaced by generalized multistar (GLM) inequalities yields the same result as the LP relaxation of the two-commodity flow formulation plus the commodity flow (CF) inequalities.
However, the size of the above two-commodity is polynomial, while the two-index relaxation has an exponential amount of constraints.

Everything taken into consideration, we decide to use the two-index flow formulation, because it performs the best and is still easy to implement.

Chapter 4

Implementation

In this chapter, we will first introduce the software we used. After that, we will briefly discuss how to change the formulations from Chapter 2 and Chapter 3 to the specific example of the online supermarket Picnic. We will then consider a small example of a ACVRP to better understand the larger instances we will consider afterwards. For the larger instances, we will consider three cases: one where all the customers are fairly grouped together in distinct clusters, one where all of the customers are relatively close to each other and one case that is in between those two.
The problems were simulated on a laptop, that has an Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz processor.

4.1 Software used

4.1.1 AIMMS

AIMMS (Advanced Interactive Multidimensional Modeling System) is a software designed to model and solve optimization and scheduling-type problems. It uses a fairly comprehensive modeling language, while simultaneously having a graphical user interface that is clear and intuitive.
While it is possible for students to obtain a free license and copy of the program, it is mainly used as a commercial solver. It is employed by companies such as *Unilever, Shell* and *Zalando.*
AIMMS is designed to be used in conjunction with solvers such as CPLEX or Gurobi.
In this thesis, AIMMS (and CPLEX) solved problems to optimality using Branch and Bound (see the intro to chapter Chapter 3) in combination with the dual simplex algorithm.
The (dual) simplex algorithm is a way to solve LP-problems. Starting from a feasible solution, it works iteratively. In each iteration the algorithm finds an extreme value of the set of solutions and checks wether it is optimal. The algorithm terminates when an optimal value is found, or when it can conclude that there is no solution to the problem.

4.1.2 CPLEX

IBM CPLEX is a commercial mathematical optimization solver, based on the programming languages Python, Java, C# and C++. CPLEX solves integer programming problems and LP

problems using Branch and Bound in combination with primal or dual versions of the simplex algorithm, or using the barrier interior point method. It is also intended for use on quadratic programming problems, and convex quadratically constrained problems using Second-Order Cone Programming (SOCP).
CPLEX is part of the standard package of AIMMS. I did not directly use CPLEX, but using the interface of AIMMS, one indirectly prompts it to use CPLEX to solve their optimization problem.
In this thesis, CPLEX uses the dual simplex method to solve the LP problems.

4.2 The ACVRP instances

4.2.1 Converting the Problem

As mentioned in Chapter 2, we need to edit our ACVRP formulations a bit to make them suit the Picnic case. More specifically, we have two capacities Q_1 and Q_2 (for cooled and uncooled goods) for each vehicle, and there are two demands d_1 and d_2 associated with each customer.
Editing the two-index flow formulation to account for this is not too hard: each set of constraints that use either the demand or the capacity are split up into two sets of constraints, one for Q_1 and d_1, and one for Q_2 and d_2. The formulation then becomes:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{4.1}$$

$$\text{subject to } \sum_{i \in V} x_{ij} = 1 \text{ for all } j \in V \setminus \{0\} \tag{4.2}$$

$$\sum_{j \in V} x_{ij} = 1 \text{ for all } i \in V \setminus \{0\} \tag{4.3}$$

$$\sum_{i \in V} x_{i0} = K \tag{4.4}$$

$$\sum_{j \in V} x_{0j} = K \tag{4.5}$$

$$u_{1_j} - u_{1_i} + Q_1 x_{ij} \leq Q_1 - d_{1_j} \text{ for all } i, j \in V \setminus \{0\} \tag{4.6}$$

$$d_{1_i} \leq u_{1_i} \leq Q_1 \tag{4.7}$$

$$u_{2_j} - u_{2_i} + Q_2 x_{ij} \leq Q_2 - d_{2_j} \text{ for all } i, j \in V \setminus \{0\} \tag{4.8}$$

$$d_{2_i} \leq u_{2_i} \leq Q_2 \tag{4.9}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j \in V \tag{4.10}$$

We edit the formulations for the relaxations in the same way.
Finding the size of the fleet K_{min} also becomes harder: we need to solve a BPP where there are two types of items, and where each container has two separate spaces, one for each type of item.

4.2.2 An example

For the sake of understanding the implementation, we will we create our own customer input:

- Depot 0: Laagraven 7, 3439 LG Nieuwegein
- Customer 1: Topaaslaan 40, 3523 AW Utrecht
- Customer 2: Slotlaan 34, 3523 HD Utrecht
- Customer 3: Loopplantsoen 51, 3523 GT Utrecht
- Customer 4: W.A. Vultostraat 41, 3523 TZ Utrecht
- Customer 5: Van Lippe Biesterfeldstraat 22, 3523 VC Utrecht

In the case of Picnic, the depot is usually relatively far away, compared to the proximity of the customers to each other. Because the customers are all relatively close to each other, it is possible for the company to serve all the customers of a delivery area within an hour, starting from the first customer, making it possible to give each customer an accurate estimate of when their groceries will arrive.
Because of this, we first determine K_{min}, the minimum number of vehicles needed to serve all customers in a route. We determine K_{min} using a modified version of the Bin Packing Problem, with bins that have two compartments with capacity Q_1, Q_2. There are n items, where item i has two components which have size d_{i_1}, d_{i_2}.
We then set $K = K_{min}$, and continue with our ACVRP.
Using google maps, we find:

Figure 4.1: The network of the depot and customers, as seen in google maps.

	0	1	2	3	4	5
0	×	4.0 km	3.4 km	3.5 km	3.4 km	3.4 km
1	4.3 km	×	0.9 km	0.85 km	1.0 km	0.45 km
2	3.2 km	0.45 km	×	0.12 km	0.55 km	0.26 km
3	3.4 km	0.40 km	0.12 km	×	0.6 km	0.55 km
4	3.2 km	1.1 km	0.8 km	0.85 km	×	0.8 km
5	3.6 km	0.45 km	0.26 km	0.3 km	0.35 km	×

Figure 4.2: Distances between the customers

We say that for the e-workers (with $Q_1 = Q_2 = 10$), the following table of demands holds:

Customer	Unooled demand	Cooled demand
0,6	0	0
1	7	1
2	5	6
3	2	8
4	3	3
5	6	2

Figure 4.3: The demands of the customers

These demands are higher than they would be in practice (a demand of 8 crates of groceries is probably uncommon), and the capacity for the vehicles is a little lower, but for this particular theoretical instance we want to avoid having less than 2 vehicles, which would trivialize the problem too much.

By solving the corresponding BPP, we see that K_{min} =min$(K_{min_{cooled}}, K_{min_{notcooled}}) = 3$.

When we solve this problem exactly, we get the following table:

	0	1	2	3	4	5
0	0	0	1	1	0	1
1	1	0	0	0	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	0	0	0	1	0

Figure 4.4: Solution to our ACVRP example

This table corresponds to having three routes:

- one route from the depot, to 2 and then back to the depot

- one route from the depot, to 3, to 1, and then back to the depot

- one route from the depot, to 5, and then via 4 back to the depot

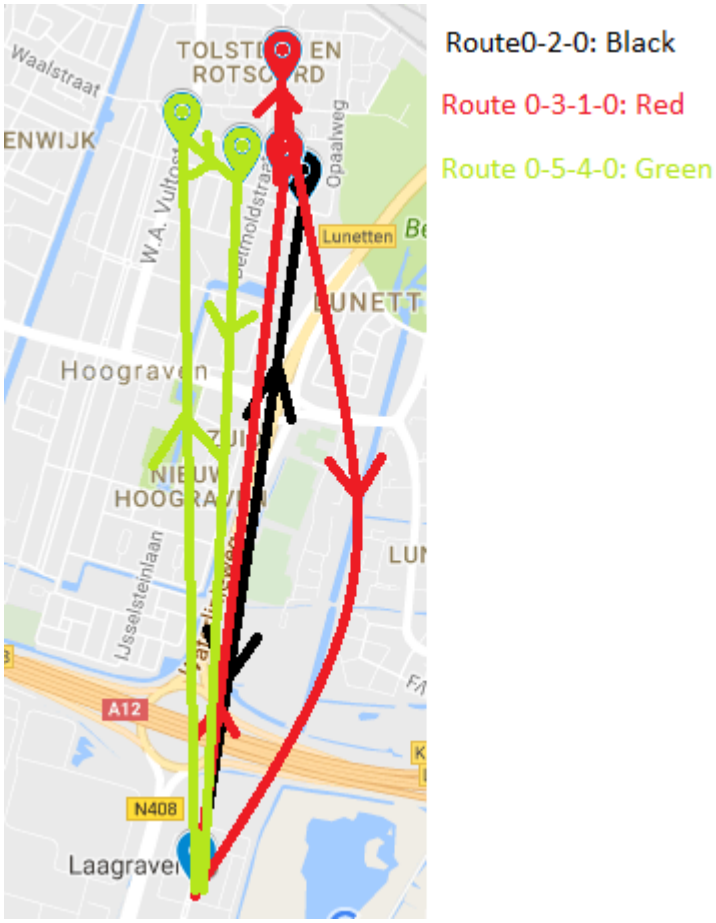

Figure 4.5: The routes illustrated on google maps

The total cost of this solution is 21.8, and it took us 0.008 seconds to solve it.
We will now take a look at the result of applying relaxations (more specifically, the AP bound and LP relaxations).

4.2.3 Applying the AP-relaxation

As previously noted, we will use the AP lower bound, as described in Section 3.4.2.
We transform the cost matrix (Figure 4.2) following the guidelines from Section 3.4.2 and obtain the following table of costs using $\lambda = 999$ as the large number used as a cost for traveling between two of the copies of the depot:

	0	1	2	3	4	5	6	7
0	×	4.0 km	3.4 km	3.5 km	3.4 km	3.4 km	×	×
1	4.3 km	×	0.9 km	0.85 km	1.0 km	0.45 km	4.3 km	4.3 km
2	3.2 km	0.45 km	×	0.12 km	0.55 km	0.26 km	3.2 km	3.2 km
3	3.4 km	0.40 km	0.12 km	×	0.6 km	0.55 km	3.4 km	3.4 km
4	3.2 km	1.1 km	0.8 km	0.85 km	×	0.8 km	3.2 km	3.2 km
5	3.6 km	0.45 km	0.26 km	0.3 km	0.35 km	×	3.6 km	3.6 km
6	×	4.0 km	3.4 km	3.5 km	3.4 km	3.4 km	999 km	999 km
7	×	4.0 km	3.4 km	3.5 km	3.4 km	3.4 km	999 km	999 km

Figure 4.6: The transformed table of travel costs for L_{AP}

After this, we remove Constraints (2.14) and (2.15), and edit Constraints (2.12) and (2.13) to include constraints for vertices 0, 6 and 7.
When we solve this new relaxed problem, we find the following:

	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1
5	0	1	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0

Figure 4.7: Solution to our relaxed ACVRP example

Which corresponds to:

- one route from the depot, to 2, to 6, to 3, and then back to the depot
- one cycle between 1 and 5
- one cycle between 4 and 7

With a total travel cost of 21.0 km, taking us 0.001 seconds to solve. Notice that the AP bound is pretty close to the actual optimal solution, within 4%.
Clearly, this is not a feasible solution for the normal ACVRP (there are two circuits disconnected from the depot, and the first route exceeds capacity for the uncooled goods).

4.2.4 The LP-relaxation

We will now apply the LP-relaxation described in Section 3.1. We only have to change the requirements for the variables x_{ij}, so we can quickly start calculating the LP bound, which gives us the following table for the x_{ij}:

	0	1	2	3	4	5
0	0	0	0.8	0.8	1	0.4
1	0	0	0.2	0.2	0	0.6
2	1	0	0	0	0	0
3	0.1	0.9	0	0	0	0
4	1	0	0	0	0	0
5	0.9	0.1	0	0	0	0

Figure 4.8: Solution to our LP-relaxed ACVRP example

With a total cost of 21.3, taking us 0.003 seconds to solve. While it took us a little longer to solve (which is of course not yet a very interesting indicator with these small instances), it produces another good bound, one of higher quality than the AP one: it is within 3% of the optimal solution value.

It is too early to draw any conclusions from these small, almost trivial instances. We will now take a look at more substantial problems, as mentioned before.

4.3 Bigger instances

Now that we have an understanding of our methods, we can take a look at some more interesting instances.
For each of the following three cases, we will consider 3 to 5 instances with 50 customers. We assume that the depot is relatively far from the customers. We will assume a homogeneous fleet with the capacity for both cooled and uncooled goods equal to 18.
Because of the relation between the amount of customers on a route and the average distance between two customers on a route (see Section 2.2, equation (2.10)), we want routes of at most 10 customers (at which point the average length between two customers becomes around 700 metres. With 11 customers this is less than 200 metres, which is not very realistic). As such, we randomly generate the demands as an integer between 1 and 4, which brings the average route to an amount of customers of about 7. See Figure 4.9 for the table of demands.
This, in combination with our capacity of 18, means that our fleet will have $K = 7$ vehicles. Note that $K_{min} = 6$, but then we would need routes of up to 10 customers, and that is not realistic with the constraint on route length.
For each customer, we let the distance to the depot be relatively large: in each of the following instances, we let the distance from any customer to the depot be a random number between 3 and 5. This is fairly realistic, as depots are often located outside of neighbourhoods, in industrial areas for example.
In the following instances, we have set the maximum amount of runtime for the algorithm to two hours.
More than this (or even just this) would take too long to be relevant for an online supermarket, and the results we do get with these approximated solution values (in essence heuristic solutions) are good enough to be drawn conclusions from.
For the specific instances, I encourage sending me an e-mail at *lennart.kerckhoffs@hotmail.com*, so I can send you the specific data for studying and reproducing.

i	$d_{cooled}(i)$	$d_{uncooled}(i)$
1	3	4
2	4	3
3	2	3
4	1	0
5	1	1
6	0	2
7	3	1
8	4	0
9	4	2
10	3	1
11	4	2
12	3	2
13	1	4
14	1	1
15	1	2
16	3	3
17	2	3
18	2	4
19	1	2
20	2	3
21	4	3
22	1	3
23	0	2
24	2	1
25	3	0
26	1	2
27	1	1
28	2	0
29	0	2
30	4	3
31	1	1
32	4	3
33	3	1
34	2	3
35	0	3
36	1	4
37	2	1
38	1	1
39	1	3
40	2	4
41	1	3
42	3	2
43	2	1
44	3	2
45	4	1
46	1	1
47	3	3
48	3	4
49	3	2
50	2	2

Figure 4.9: The demands of our customers in the coming problems, generated using ASELECT in Excel

4.3.1 The "Ideal" Case

To have something to compare the other results to, we will now generate an (in theory) ideal case, for which we should be able to solve the problem very quickly.
In this ideal case, we group customers together in clusters in such a way that one vehicle is sufficient to serve all customers in that cluster. The distance between two customers that are in different clusters will be generated as a random number between 2 and 5.5 kilometres, while the distance between two customers within the same cluster will be between 0.2 and 1.5 kilometres. We will ignore the constraint of route length here, assuming that the customers within one route are sufficiently close to one another that there will not be any problems.
This way, solving this instance of the ACVRP reduces (for a human, at least) to solving 6 ($K_{min} = 6$, as mentioned before) separate TSP's. While this is still an NP-hard problem, 6 TSP's with 6 to 12 customers are in theory a lot easier to solve than one ACVRP with 50 customers and 6 vehicles.
Notice that this is not, in practice how we actually solve this problem. We do however "hope" that our solving software is able to recognize this problem as being easily solvable from the data it is provided.
The tables for this data were judged to large to be included in this thesis. AIMMS was able to solve this instance to optimality in 83.945 seconds, and returned an optimal solution value of 67.2 km.
The AP relaxation solution had a value of 20.3 after 0.080 seconds, and the LP relaxation had a value of 62.8, taking us 0.074 seconds. We cannot yet draw conclusions about this instance, as we have not yet checked the other instances. We can, however, already see that the AP relaxation, while it is very fast, gives us a terrible lower bound: within 70% is a way too large interval to gain any useful information about the quality of a potential heuristic solution. On the other hand, the LP solution is also found very quickly, and gives a value within 7% of the optimal solution value. While still farther off than we would want, it is a more useful relaxation than the AP relaxation.

4.3.2 Clustered Customers

In our first case, we assume that the customers are close together. In theory, this might make the problem easier to solve, if the software we use is able to detect that certain groups of customers that are close to each other can be served by a subset of the vehicles (just like in Section 4.3.1). In practice, this is not how we will solve this problem, because sometimes a vehicle needs to deliver groceries to multiple clusters to use the vehicles and their capacity more efficiently.
When looking at clustered customers, we can assume that the total amount of customers in a route is not as important as the total amount of customers in a route in one cluster with regards to the maximal route length. This is because the supermarket can make sure their customers are scheduled in such a way that the 1-hour time window will not pose any problems for vehicles visiting two different customer clusters.
We will generate the distance between two customers in the same cluster as a random number between 0.2 and 1.5 kilometres. The distance between two customers of different clusters will again be generated as a random number between 2 and 5.5 kilometres.
We want each cluster to be able to be served by 1 or 2 vehicles. As such, we want around 7 or 14 customers in each cluster.
We generate each cluster in the following way:

- randomly let the cluster be able to be visited by either 1 or 2 vehicles (approximately). We call this number k.
- let the size of the cluster be a random integer generated between $(7 \cdot k) - 3$ and $(7 \cdot k) + 3$.
- randomly determine which customers will be in the cluster.

We expect these instances to be the most easy to solve, because of the problem being nearly separable into smaller problems (with a lot less possible solutions in total).
We generated three cases: the first had 4 clusters, the second 5 clusters and the last 6 clusters.

- AIMMS was able to solve the first instance to optimality in 380.34 seconds, and returned an optimal solution value of 63.2 km. The AP relaxation solution had a value of 59.3 after 0.082 seconds, and the LP relaxation had a value of 59.3, taking us 0.06 seconds.
- AIMMS was able to solve the second instance to optimality in 399.38 seconds, and returned an optimal solution value of 65.8 km. The AP relaxation solution had a value of 62.4 after 0.08 seconds, and the LP relaxation had a value of 62.4, taking us 0.085 seconds.
- AIMMS was not able to solve the third instance to optimality, but it did manage to find a feasible solution in 7202.2 seconds, and returned a solution value of 67.9 km. The AP relaxation solution had a value of 62.5 after 0.079 seconds, and the LP relaxation had a value of 62.6, taking us 0.078 seconds.

Compared to the "ideal" case, the problems were a lot harder to solve (one instance even had to be terminated early).
The relaxations, however, did a little better in general. Note that the LP and AP relaxations gave (almost) the exact same quality of the bound, while the LP relaxation was a little faster (however, negligibly so).

4.3.3 All customers relatively close

When all customers are relatively close to each other, such that we cannot cluster different groups of customers, we expect the computation time to be a little longer, as we need to consider more options when assigning customers to the vehicles. We will generate the distance between any two customers as a random number between 0.2 and 1.5 kilometres, as with the customers in the same cluster in Section 4.3.2 (essentially, we have instances with one big cluster of customers here).
We expect these instances to take the longest to be solved, because the algorithm essentially needs to consider all options, without being able to quickly terminate possible solutions.
We generated 3 different instances:

- Instance 1 took us 34.492 seconds to solve, and returned an optimal solution value of 56,4. The AP relaxation solution had a value of 54.7 after 0.082 seconds, and the LP relaxation had a value of 54.7, taking us 0.075 seconds.
- AIMMS was able to solve the second instance to optimality in 168.94 seconds, and returned an optimal solution value of 56.3 km. The AP relaxation solution had a value of 54.1 after 0.079 seconds, and the LP relaxation had a value of 54.1, taking us 0.068 seconds.

- AIMMS was able to solve the second instance to optimality in 2.74 seconds, and returned an optimal solution value of 56.8 km. The AP relaxation solution had a value of 10.2 after 0.081 seconds, and the LP relaxation had a value of 55.3, taking us 0.097 seconds.

Contrary to our expectations, the cases where the customers are not divided into separate clusters is easier to solve for AIMMS than the case where the customers are clustered. It might even be faster than what we considered to be the "ideal" case. This could possibly be because in the clustered case, customers might "accidently" be clustered in such a way that almost every vehicle needs to visit more than one cluster. We can see this reflected in the costs too: the optimal solutions for the clustered cases are higher than for the not clustered cases: the extra costs can be expected to come from travel between two different clusters. Because of this traveling in between clusters, the algorithm has to consider not only each customer that will minimize the cost of the route (as with the not clustered case), but before that which cluster constitutes the best possible choices of customers.
The AP relaxation has again a case where it fails and gives a fairly useless lower bound of 10.2, compared to the optimal value of 56.8. Note that this third case was solved extremely quickly, in 2.74 seconds. In the other cases, the LP and AP relaxations performed comparably, and gave fairly decent bounds, within 4% of the optimal solution.

4.3.4 Combination of the previous two cases

In these instances, we will generate instances with two small clusters (of 5 to 7 customers, as in Section 4.3.2), and group all other customers together in one big cluster.
We generated 4 instances, two with two clusters aside from the on big cluster, and two with three clusters aside from the one big cluster.

- Instance 1 took us 795.95 seconds to solve, and returned an optimal solution value of 76,9. The AP relaxation solution had a value of 72.8 after 0.03 seconds, and the LP relaxation had a value of 72.8, taking us 0.085 seconds.
- Instance 2 took us 447.87 seconds to solve, and returned an optimal solution value of 62.3. The AP relaxation solution had a value of 57.9 after 0.064 seconds, and the LP relaxation had a value of 57.9, taking us 0.079 seconds.
- AIMMS was not able to solve the third instance to optimality, but it did manage to find a feasible solution in 7202.02 seconds before calculations were terminated, and it returned a solution value of 63.5. The AP relaxation solution had a value of 59 after 0.071 seconds, and the LP relaxation had a value of 59.1, taking us 0.041 seconds.
- AIMMS was not able to solve the fourth instance to optimality, but it did manage to find a feasible solution in 7206.7 seconds before calculations were terminated, and it returned a solution value of 64.9. The AP relaxation solution had a value of 60.1 after 0.079 seconds, and the LP relaxation had a value of 60.1, taking us 0.079 seconds.

Again we are a little surprised by the results: this case seems to be the most computationally difficult of all the cases overall. Looking at the large costs, this might again be a problem caused by the need to travel in between clusters alot, as with the clustered case (see Section 4.3.3).
The computation time and quality of the bound of the AP and LP relaxations were again

comparable, with speed being around the same on average, but the LP relaxation winning out with the quality of the bound. In general, the bounds were within 7% of the optimal solution value, which is okay.

Chapter 5

Conclusions and reflections

In this thesis, we took a look at the Asymmetric Vehicle Routing Problem. We introduced different ways to model this problem mathematically. Taking the speed of solution methods, possible relaxations and, sadly, time constraints into account we opted to use a Two-Index Flow formulation to model the problem.
We then looked at different possible relaxations for the problem, and chose to apply both a Linear Programming relaxation and an Assignment Problem relaxation.
In Chapter 4, we extended the ACVRP to our specific example, where we had vehicles with two different demands: one for cooled, and one for uncooled goods.
After that, we took a look at 4 different cases to test our model and relaxations from the previous chapters in practice. These cases were randomly generated. For each case and each instance of a case, the same table of demands for the customers.
We saw that, contrary to what we expected, cases where all of the customers are grouped relatively close together, the problem is more easy to solve than when the customers are divided into separate clusters, with larger distances between customers in separate clusters.
Also, while ACVRPs with a large number of vehicles and thus a large number of routes (due to a low capacity per vehicle) are computed faster than ACVRPs with relatively larger capacity.
We also saw that the LP relaxation works better than the AP relaxation in quality of the bound while requiring around the same computation time. It should be mentioned that the AP relaxation also requires a bit more work before it can be applied in each case, because the graph has to be extended depending on the number of vehicles used. We also saw that the AP relaxation "flops" in rare cases, providing us with near useless lower bounds.

This research is by no means finished: we did not have the chance to test too many different instances for each case, so our conclusions might be a little unreliable, or off.
Due to time constraints, we also neglected some possible relaxations, such as the Lower Bound Based on Flow and the Disjunctive Lower Bound. While these require more extensive algorithms, they should in theory provide high quality bounds. Aside from that, more formulations of the ACVRP can be tested. For example, one could see if a Set Partitioning formulation yields better results in solving the problem exactly, or has better relaxations for finding lower bounds on the optimal solution value.
Lastly, everything mentioned above could be applied to even bigger instances with more customers, or problems where the fleet is non-homogeneous, or at least the capacity for cooled and uncooled goods is different.

Bibliography

[1] Roberto Baldacci, Paolo Toth, Daniele Vigo *Recent advances in vehicle routing exact algorithm.* `http://dx.doi.org/10.1007/s10288-007-0063-3`

[2] Paolo Toth, Daniele Vigo *Models, relaxations and exact approachtes for the capacitated vehicle routing problem.* Università di Bologna, Dipartimento die Elettronica, Informatica e Sistemistica, Viale Risorgimento 2, 40136 Bologna, Italy

[3] A.H. Escobar, Z.M. Granada E. E.M. Toro O,. *Literature review on the vehicle routing problem in the green transportation context.* Revista Luna Azul, 42:362-387, 2016.

[4] G. W. Klau *Lagrangian Relaxation: An Overview* Discrete Mathematics for Bioinformatics WS 07/08, 18^{th} of december 2007

[5] H. Yeo *Course notes* IEOR 251 - Facility Design and Logistics, March 16, 2005

[6] M. Held, R.M. Karp *The traveling salesman problem and minimum spanning trees: Part II*, Math. Programming 1 (1971)

[7] Finke G, Claus A, Gunn E (1984) *A two-commodity network flow approach to the traveling salesman problem.* Congress number 41: 167-178

[8] M.L. Balinski and R.E. Quandt *On an integer program for a delivery problem* Princeton Universtiy, Princeton, New Jersey March 11, 1963

[9] Christos H. Papadimitriou, Kenneth Steiglitz *Combinatorial Optimization: Algorithms and Complexity* 1982

[10] L.A. Wolsey *Integer Programming* December 1998

[11] I. Kara, G. Laporte, T. Bektas *A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem* Received 30 September 2002; accepted 5 May 2003

[12] Miller, C. E.; Tucker, E. W.; Zemlin, R. A. (1960). *Integer Programming Formulations and Travelling Salesman Problems* J. ACM. 7: 326–329

[13] M. Fischetti (University of Padova, Padova, Italy), P. Toth, D. Vigo (University of Bologna, Bologna, Italy) *A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs* Received August 1992; revision received may 1993; accepted June 1993

[14] David L. Applegate, Robert E. Bixby, Vasek Chvátal and William J. Cook *The Traveling Salesman Problem: A Computational Study* Princeton University Press, 2006

[15] M. Grötschel, L. Lóvasz, A. Schrijver *Geometric Algorithms and Combinatorial Optimization* Technische Universität Berlin, Eötvös Loránd University, University of Amsterdam 1988