A Column Generation Heuristic for Multi-Depot Vehicles Scheduling with Electric Vehicles

Jim van Houwelingen - 373537

Master Thesis in Econometrics and Management Science
Specialization: Quantitative Logistics and Operations Research

Abstract

In this thesis we have formulated a method to solve the Multi-Depot Vehicle Scheduling Problem with a bus-fleet which is partially electrically fuelled. This method incorporates a Column Generation process and a dedicated heuristic to solve the Pricing Problem. The Truncated Column Generation heuristic provides integer solutions in reasonable time for large instances of the MD-VSP. Two additional challenges surface with the introduction of electric vehicles: The limited action radius of the batteries, resulting in a more challenging Pricing Problem. Secondly, electric vehicles are smaller than diesel vehicles in terms of passenger capacity. To incorporate the electric fleet, we have introduced a new set of arcs to the network for which a bus can be recharged. Additionally, we have determined the demand per trip and removed arcs for which a bus as insufficient capacity. The algorithm used to solve the Pricing Problem is similar to any shortest path algorithm. We introduced one additional update-condition to ensure there is always sufficient charge left in battery to continue and return to the depot to recharge. The heuristic is used to solve the vehicle scheduling problem based on historical data for trip demand. We have compared the results of these schedules to the schedules we have solved for each weekday for two test weeks. We conclude that the on average the operational costs using day-specific schedules are reduced by 9.3%.

ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics
Supervisor: Dennis Huisman
Second Assessor: Sharif Azadeh
Final Date: October 16, 2018

CONTENTS

1 Introduction 3
2 Literature Review 5
2.1 Vehicle Scheduling Problem 5
2.2 Connection-based Formulation 6
2.3 Path-based Formulation 9
2.4 Electric Vehicle Scheduling Problem 11
2.5 Stochastic/Dynamic VSP 11
3 Problem Description 13
3.1 Concession 13
3.2 Fleet 14
3.3 Regulations for Capacity 15
3.4 Restrictions for Electric Vehicles 16
3.4.1 Assumptions 16
3.4.2 Recharge-function 16
4 Mathematical Formulations 18
4.1 General Notation 18
4.2 The Network 18
4.2.1 Recharge Arcs 19
4.2.2 Occupancy rate 19
4.3 Set-Covering Formulation 20
5 Solution Approach 22
5.1 Restricted Master Problem 22
5.2 The Pricing Problem 22
5.3 Dedicated Algorithm for Pricing Problem 23
5.4 Proof of Correctness 27
5.5 Truncated Column Generation Heuristic 29
6 Data 31
6.1 Alterations to the data-set 31
6.2 Occupancy Rate 31
6.3 Fitting Distributions 32
6.4 Test Weeks 34
7 Results & Discussion 35
7.1 Computer Specifications 35
7.2 Initialization 35
7.3 Schedules Based on Historic Demand 36
7.3.1 Heuristic Performance 36
7.3.2 The Solution 38
7.4 Day-specific Schedules 40
7.5 Discussion of the Results 42
8 Conclusion & Future Research 44
9 Appendix 46
9.1 A: Buslines 46
9.2 B: Recharge Function 48

1 INTRODUCTION

Over the past few decades, much effort has been put into investigating methods to solve large Vehicle Scheduling Problems. Currently large problems for aircraft-carriers, train-transportation and bus-transit have to be solved in order to remain economically viable. Bus-transit companies in the Netherlands are contractually bounded to provide public transportation for certain regions or concessions. However, acquiring these contracts is competitive. Therefore, to remain competitive it is necessary for those companies to be as cost efficient as possible. The focus of this thesis is specifically on public bus transportation.

The planning process for public transportation consists of four steps. In step one, the timetables consisting of all trips has to be made. This task, while maximizing customer satisfaction and minimizing costs, is a very complex problem. Secondly, the public transport company has to schedule which bus is going to perform which trip. With various types of buses, route- and time-constraints minimizing costs becomes increasingly difficult. Next, anonymous duties are scheduled specifically stating at which time on which bus line(s) someone is going to work. Lastly the company has to determine who is going to perform a specific duty: assigning a bus driver to each duty.

While every step in the planning process is an interesting problem by itself, this thesis will exclusively focus on step 2: scheduling the available bus-fleet optimally to a given timetable for region Noord-Holland Noord which is carried out by bus transit company Connexxion. For the past few decades the Vehicle Scheduling Problem has been researched intensively. The problems become larger and larger, therefore using the available resources efficiently becomes ever so important. The latest developments in the field of Vehicle Scheduling are focused on solving the instances faster using heuristics and simultaneously solving the scheduling problem for both the vehicles and crews.

Another subject of interest is the deployment of electric vehicles. Based on international regulations, transportation companies are obliged to reduce carbon emissions. Therefore the bus-fleet of public transit companies is slowly becoming more and more electric. Contrary to the conventional bus, the batteries of an electric bus have a limited range before it has to return to a depot or charging station to recharge. In large cities, diesel engines are prohibited to enter city centres, which imposes a restriction on which type of bus to use. In addition to having a smaller action-radius compared to the conventional bus, the electric vehicles are often smaller in terms of passenger capacity.

A recent solution to deploy buses efficiently in terms of capacity is of a more practical nature: public transportation companies have introduced demand-driven transportation. This means a bus will only be deployed when customers make their demand known. However, this only happens for low-demand trips - trips which are likely to have zero demand. Having multiple types of buses with a different capacity, range and cost and not knowing the demand for a trip, the choice for a type of bus is relevant for all trips. A bus transit company cannot leave anyone behind at a bus-stop. Therefore it needs sufficient capacity at all times. Using smaller buses is cheaper, but bears the risk that the total demand for a trip exceeds its capacity. The new challenge is to schedule as efficiently as possible, while satisfying all demand.

The main purpose of this thesis is to formulate a method to find an optimal schedule for a given timetable using a fleet which is predominantly electric. In this optimal schedule we want to guarantee that there is sufficient capacity at any point of time. To do so, the following research questions are defined:

- How to incorporate the passenger demand in the network?
- How to incorporate the range limitation of the electric vehicles in the heuristic?

The public transport company is currently using historical demand to decide whether using a small or larger bus. A particularly interesting question to ask is what value a forecasting tool has, which can predict the maximum occupancy rate of a trip. We are interested in what the benefits are of a proper forecasting model to predict the number of passengers on a certain trip. Therefore we define the final research question:

- What is the benefit of making a vehicle schedule every day using the forecasts for trip occupancy?

To be able to answer these questions, we start with an outline of this thesis. In chapter 2 we will elaborate on the relevant literature for Vehicle Scheduling Problem (VSP), with special attention for the Multi-Depot Vehicle Scheduling Problem (MDVSP) and Electric Vehicle Scheduling Problem (E-VSP). With this review, a clear set of approaches to evaluate this problem becomes available. In chapter 3 the case provided by Connexxion is introduced and described in detail. We will introduce the mathematical formulation and define the network in chapter 4. In chapter 5, we will discuss the solution approach. Chapter 6 is dedicated to the occupancy rate for all trips which have to be performed. The results are shown and discussed in chapter 7. Finally, in chapter 8 this thesis will provide a conclusion based on its findings.

2 LITERATURE REVIEW

In this chapter we review the literature considering the Vehicle Scheduling Problem. First we elaborate on the Vehicle Scheduling Problem in general. To continue, an explanation is given for two approaches to solve the Multi-Depot Vehicle Scheduling Problem. Lastly we summarize the literature on the Electric Vehicle Scheduling Problem and the Vehicle Schedule Problem with stochastic elements.

2.1 Vehicle Scheduling Problem

Due to the complexity and the large instances which have to be solved in the real world, there is a lot of literature available for solving the MDVSP. Surveys on the VSP (Daduna and Paixão 1995, Kliewer and Bunte 2009) provide several variations and special cases of VSP which makes for an excellent starting point for this thesis.

The aim of the Vehicle Scheduling Problem can be narrowed down to three characteristics:

- Each trip is covered exactly once
- Each vehicle performs a path from depot to depot
- Vehicle cost is minimized

A **'trip'** is defined as a route from a start location to an end location covering all intermediate stops. Each trip is defined in a timetable, stating a starting location l^s, end location l^e and starting time t^s and ending time t^e. Besides the scheduled trips, there are several types of trips to link depots to trips which do not carry any passengers. Travelling from a depot to a scheduled trip is called a **'pull-out trip'** and travelling from a scheduled trip to a depot is called a **'pull-in trip'**. 'Dead-heading' or a **'dead-head trip'** is sending a bus from location A to location B without it being in service. Define τ_{ij} the dead-head (travel) time from end-location trip i to start-location trip j. Trip j is **'compatible'** with trip i, when $t^s_j \geq t^e_i$ and $l^s_j = l^e_i$ holds, or $t^s_j \geq t^e_i + \tau_{ij}$ where $l^s_i \neq l^e_j$. Stating that trip j is compatible with trip i the notation $i \; \alpha \; j$ is used. A **'path'** is defined as a sequence of compatible trips performed consecutively starting from the depot and ending at the depot.

The objective function generally consists of two parts. The *fixed* costs of using a bus and the *operational* or variable costs. The fixed costs mostly consists of investment costs and maintenance of buses. Operational costs often consists of fuel-costs and maintenance costs, which may simply be expressed as cost per kilometer travelled. Typically, the combination of vehicles used and total (dead-head) distance travelled is minimized. Though, the objective function may serve many purposes.

The most prominent difference in literature among instances of VSP is the number of depots present in the network. In particular whether there is a single depot or multiple depots. It is well-known that the

VSP with a single depot is equivalent to the Min-Cost-Flow-Problem, which is solvable in polynomial time. Formulations with multiple depots (MDVSP) require an additional set of constraints to cover all trips exactly once. MDVSP is $\mathcal{NP}$-hard: a formal proof is shown in (Bertossi, Carraresi, and Gallo 1987). The complexity of each instance depends on the number of trips to be performed, the number of depots and vehicle types available and the number of arcs in the network.

Predominantly, optimization models are based on the Network Flow Problem as it is the more intuitive way to solve the MDVSP. There are two approaches using the Network Flow Problem which we will investigate in this thesis:

- connection-based approach
- path-based approach

Both approaches will be treated and evaluated separately. Starting with the connection based approach and finishing with the path-based approach. Both approaches will emphasize on the Multi-Depot Vehicle Scheduling Problem.

2.2 Connection-based Formulation

There are several different formulations for the MDVSP, though the most straightforward formulation is given in formulation (M1). The following notation is introduced: When speaking of a graph or network we will use the notation $G = (V, A)$, where V is the set of all vertices and A the set of all arcs in the network. Furthermore, let the following sets be defined such that:

- N: the set of all timetabled trips to be performed
- $\mathcal{D}$: is the set of all depots
- $\mathcal{D}(i)$: the set of depots from which trip $i \in N$ can be performed
- $N(d)$: The set of trips that can be performed by vehicles from depot $d \in \mathcal{D}$

Introduce the following variables and parameters: Denote x_{ij}^d the flow of vehicles from depot d over arc $(i, j) \in A$. Let $y_i^d = 1$ if trip i is assigned to a vehicle from depot d, 0 otherwise. Denote c_{ij}^d the cost per vehicle from depot d of using arc $(i, j) \in A$ and lastly u^d the number of vehicles present at depot d.

$$(M1) : min \sum_{d \in \mathcal{D}} \sum_{(i,j) \in A} c_{ij}^d x_{ij}^d \tag{2.1}$$

$$s.t \sum_{j:(i,j) \in A} x_{ij}^d = y_i^d \qquad \forall d \in \mathcal{D}, i \in N(d) \tag{2.2}$$

$$\sum_{i:(i,j) \in A} x_{ij}^d = y_j^d \qquad \forall d \in \mathcal{D}, j \in N(d) \tag{2.3}$$

$$\sum_{d \in \mathcal{D}(i)} y_i^d = 1 \qquad \forall i \in N \qquad (2.4)$$

$$\sum_{i \in N} y_i^d \leq u_d \qquad \forall d \in \mathcal{D} \qquad (2.5)$$

$$y_{ij}^d \in \{0,1\} \qquad \forall d \in \mathcal{D}, (i,j) \in A \qquad (2.6)$$

$$x_i^d \in \{0,1\} \qquad \forall d \in \mathcal{D}, i \in N \qquad (2.7)$$

Constraint sets (2.2) and (2.3) ensure that the flow is conserved and feasible. Constraints (2.4) make sure that each trip is assigned to a vehicle from depot d exactly once. With constraint set (2.5) the number of vehicles present in the network cannot be exceeded. Constraints (2.6) and (2.7) are domain constraints.

As stated before, the real-life instances of the VSP can be extremely large and therefore are too time-consuming to solve to optimality with 'weak' formulations. Therefore, there is a need to reformulate the problem. The Multi Depot Vehicle Scheduling Problem is equivalent to the Multi-Commodity Flow Problem. To illustrate, a Single-Commodity Flow Problem is shown in Figure 1.

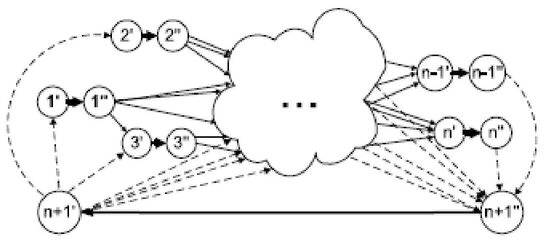


Figure 1: Single-Depot VSP network with dual nodes

Note that for each trip the nodes are duplicated, a departure-node and an arrival-node are explicitly defined for each trip. Resulting in an additional set of arcs AT between departure-nodes and arrival-nodes. Also, a circulation arc is included in the network, from the end depot to the start depot. To continue to reformulate the MDVSP into a Multi-Commodity Flow Problem a network-layer similarly to Figure 1 is defined per depot/vehicle type: $G^d = (V^d, A^d)$, where V^d are all the nodes in the network layer and A_d are all the arcs in the network-layer. This results in a multi-graph network as shown in Figure 2. Let $AT^n \subseteq AT$ be the set of all arcs corresponding to trip $n \in N$ and let AC be the set of circulation arcs of G, where $AC^d \subseteq AC$ is the set of circulation arcs for each network layer. Which, in this case, is just one arc for each depot.

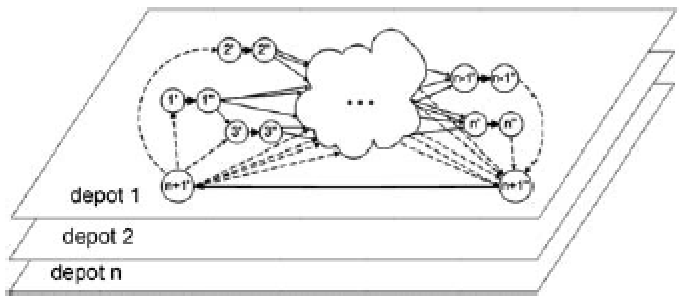


Figure 2: MDVSP network

Given this new notation for the Multi-Commodity Flow Problem, we also have to rewrite the formulation. The corresponding formulation is stated below:

$$(M2) : min \sum_{d \in \mathcal{D}} \sum_{(i,j) \in A^d} c_{ij}^d x_{ij}^d \tag{2.8}$$

$$s.t. \sum_{j \in V^d : (i,j) \in A^d} x_{ij}^d - \sum_{j \in V^d : (j,i) \in A^d} x_{ji}^d = 0 \quad \forall i \in V^d, d \in \mathcal{D} \tag{2.9}$$

$$\sum_{d \in \mathcal{D}} \sum_{(i,j) \in AT^n} x_{ij}^d = 1 \quad \forall n \in N \tag{2.10}$$

$$\sum_{(i,j) \in AC^d} x_{ij}^d \leq u_d \quad \forall d \in \mathcal{D} \tag{2.11}$$

$$x_{ij}^d \in \{0,1\} \quad \forall (i,j) \in A^d \backslash AC^d, \forall d \in \mathcal{D} \tag{2.12}$$

$$x_{ij}^d \in \mathbb{N}_+ \quad \forall (i,j) \in AC^d, \forall d \in \mathcal{D} \tag{2.13}$$

Constraint (2.9) is the flow conservation constraint. Constraint (2.10) ensures each trip is covered exactly once. Constraint (2.11) makes sure the solution found satisfies the capacity constraint for each depot d.

As stated before, the Multi-Commodity Flow problem is equivalent to the Multi-Depot Vehicle Scheduling problem. However, for formulation (M1) we had to introduce a second set of decision variables while the formulation (M2) only needs one set of decision variables.

Keeping in mind that the formulation only considers either depots or vehicle types (Ferland and Michelon 1988). Therefore, the idea of creating a network layer for each vehicle type/depot combination, resulting in $|\mathcal{D}| \cdot |\mathcal{T}|$ network-layers, as displayed in (Gintner, Kliewer, and Suhl 2005) can be used to solve the Multiple Vehicle Types MDVSP (MVT-MDVSP).

The connection-based formulation determines the flow of each type of vehicle over an arc. Solving this problem generates flows over arcs, denoting the number of vehicles using this arc, implicitly creating paths for a vehicle. Typically, each arc incorporates a cost corresponding to an activity. For example, an arc from a depot to a trip contains the start-up cost, while an arc from a trip to another trip incorporates trip-cost and dead-head cost. The objective of the VSP is to minimize the total cost incurred.

The main disadvantage of using the arc-based formulation is the number of arcs growing quadratically when the number of trips increases. In general, unrestricted dead-heading is allowed for public bus transportation as opposed to aircraft scheduling. Since the instances grow quadratically with the number of trips, several papers are dedicated to reduce the number of dead-head arcs. Assuming that morning-trips will not be followed up with either midday-trips and night-trips, will reduce the number of variables needed by 40% (Haghani and Banihashemi 2002).

The time-space formulation as described in (Kliewer, Mellouli, and Suhl 2006), aggregates the group of compatible arcs into a single arc, drastically reducing the number of arcs by 97%.

2.3 Path-based Formulation

Let $\mathcal{P}_d$ be the set of all feasible paths departing from depot d. Let $z_p \in \mathbb{B}$ be the choice whether the path p is being used or not. Define parameters a_{jp} whether trip j is included in path p. Define parameter r_d being the maximum capacity of depot d. State c_p the costs included to use path p. This results in formulation (M3) as stated below:

$$(M3) : min \sum_{d \in \mathcal{D}} \sum_{p \in \mathcal{P}_d} c_p z_p \tag{2.14}$$

$$s.t. \sum_{d \in \mathcal{D}} \sum_{p \in \mathcal{P}_d} a_{jp} z_p = 1, \qquad \forall j \in N \tag{2.15}$$

$$\sum_{p \in \mathcal{P}_d} z_p \leq r_d \qquad \forall d \in \mathcal{D} \tag{2.16}$$

$$z_p \in \{0, 1\} \qquad \forall d \in \mathcal{D}, \forall p \in \mathcal{P}_d \tag{2.17}$$

Constraints (2.15) ensure every trip is performed exactly once. Constraint set (2.16) makes sure that the number of buses used per depot does not exceed the capacity of each depot. Concluding with the integrality constraint (2.17). The objective function minimizes the total costs. In this case, all costs corresponding to path p are implied in c_p and may have various goals. For example, setting $c_p = 1, \ \forall p \in \mathcal{P}$ would minimize the number of paths used and therefore minimizing the fleet-size required to solve the problem.

The main advantage of using the path-based formulation is that the notation is intuitively easy to grasp. A further advantage is that fuel-constraints or time-constraints for vehicles or corresponding duties are easily recognized for paths compared to the connection-based models. However, the number of feasible paths grows exponentially when an instance increases in size, making it highly inefficient/impossible to enumerate every single possible path. Nonetheless, starting with a primal feasible set of circuits in combination with column generation (as shown in Ribeiro and Soumis 1994) is a strong approach to solving the MDVSP.

Column Generation

Starting with the set-covering formulation (M3), column generation is applied to solve the MDVSP to optimality. For an in-depth overview of the derivation of the Column Generation formulation, we refer to (Desrosiers and Lübbecke 2005). Introducing dual variables μ_j for all $j \in N$ and λ_d for all $d \in \mathcal{D}$ associated to the set of constraints (2.15) and (2.16), respectively. The corresponding pricing problem for column generation is a Shortest Path Problem as shown in formulation (M4) for each depot d:

$$(M4) : min - \lambda_d + \sum_{(i,j)\in A^d} (c_{ij} - \mu_j) x^d_{ij} \tag{2.18}$$

$$s.t. \sum_{i\in V^d} x^d_{ij} - \sum_{i\in V^d} x^d_{ji} = 0 \qquad \forall j \in N \tag{2.19}$$

$$\sum_{i\in N} x^d_{n+d,i} = 1 \tag{2.20}$$

$$\sum_{i\in N} x^d_{i,n+d} = 1 \tag{2.21}$$

$$x^d_{ij} \in \{0,1\} \qquad \forall (i,j) \in A^d \tag{2.22}$$

An optimal solution for the Shortest Path Problem: $\bar{x}^d_{ij}$ for all $(i,j) \in A^d$, is a feasible path from depot to depot. To translate this solution to a path which can be incorporated in the set covering formulation, we use $c_p = \sum_{(i,j)\in A^d} c_{ij}\bar{x}_{ij}$ and $a_{jp} = \sum_{i\in V^d} \bar{x}^d_{ij}$ for all $j \in N$. When the solution value corresponding to $\bar{x}_{ij}$ is less than 0, there are reduced costs. Implying when this path is added to $\mathcal{P}$ a better solution may be found. When there are no reduced costs, no improvement can be found. Therefore, the problem is solved to optimality. (Oukil, Amor, and Desaulniers 2006) proposed a stabilized column generation method which handles highly degenerate instances of Vehicle Scheduling Problems efficiently.

There are several heuristics to find a proper initial solution which provides the primal base set of paths to perform column generation. There is a trade-off between computation-time and optimality-gap using a heuristic to find a feasible solution. The faster an initial feasible solution is found, the earlier we can start the column generation process. However, starting with a better initial feasible solution, the column generation process may find the optimal solution faster. Since the MDVSP has been investigated for over 30 years, many heuristics have been developed. Simple ideas such as creating disjoint subsets of trips which are to be covered by a single depot, creates several SD-VSP problems which can be solved in polynomial time. (Carrareri and

Gallo 1984). More sophisticated heuristics are compared in (Pepin et al. 2009) including Lagrangian heuristics and Tabu-Search among others.

2.4 Electric Vehicle Scheduling Problem

The introduction of full-electric vehicles has led to a new challenge for public transit companies. Due to the battery capacity the range of an electric vehicle is severely limited compared to the conventional diesel buses. Therefore, there is a need to incorporate these limitations in our formulation for electric vehicles. Recent efforts to solve the VSP with range constraints are discussed in (Haghani and Banihashemi 2002). (Wang and Shen 2007) propose a heuristic based on an Ant Colony Optimization method (ACO). Though, both methods require to recharge at a depot. There are many assumptions made to reduce the complexity, especially for the charging method and recharging of a battery. (Chao and Xiaohong 2013) allow for swapping batteries at depot which would imply a constant recharging-time. However, swapping batteries is not applicable for many types of buses. Similar to the assumption of constant recharging time is when the infrastructure present in the network allows for fast-recharging. Slow recharging is much more complicated, as it requires you to explicitly keep track of the battery charge. The time required to recharge the battery cannot be ignored. Also the decision to what extend the battery has to be recharged is important. (Kooten-Niekerk, Akker, and Hoogeveen 2017) provide two methods to solve the E-VSP with continuous and discrete states of battery charge. Taking continuous state of charge into account is only applicable for small instances of E-VSP (< 10 vehicles). For discrete state of charge, each trip-node is duplicated for various values of battery-charge. The results shown are solved for problems with less than 800 trips using column generation. (Adler and Mirchandania 2017) proposes a Concurrent Scheduling heuristic solving the E-VSP, introducing nodes in the network representing fuel-stations. While this heuristic does not provide an optimal solution, it returns a solution very quickly.

Closely related to the E-VSP is the Electric Vehicle *Routing* Problem (VRP). (Schneider, Stenger, and Goeke 2014) and (Bruglieri et al. 2015) propose a method to solve the VRP problem with Time Windows and Route Constraints.

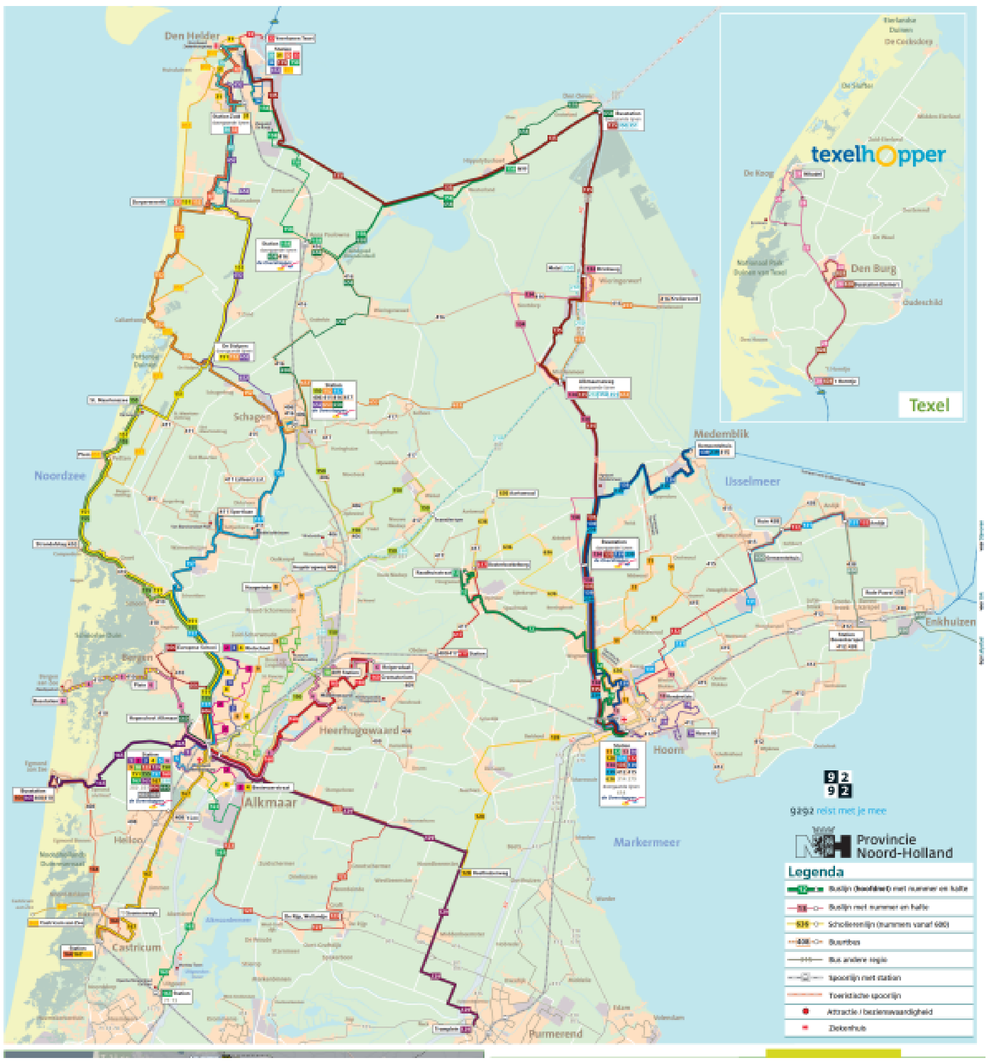
2.5 Stochastic/Dynamic VSP

So far, we have assumed that every parameter in the formulation is deterministic. However several elements in our model are stochastic in nature such as demand for a trip or natural events such as traffic jams or the breakdown of a bus during its path. Most research has been dedicated to the fluctuation of *actual* trip times. (Huisman, Freling, and Wagelmans 2004) provide a robust solution taking several scenarios for trip times into account. (Naumann, Suhl, and Kramkowski 2011) introduce a penalty for waiting times in a system with disruptions using stochastic programming.

3 PROBLEM DESCRIPTION

In this chapter we will elaborate on the problem and explain its specifics. First we will discuss the concession, followed by the fleet and capacity regulations. Lastly we will discuss the restrictions specifically for electric vehicles.

3.1 Concession



The concession Noord-Holland Noord consists of 4 regions defined by depots located in Alkmaar, Hoorn, Den Helder and Texel. Since Texel is a small island with only a ferry as a connection to the main concession, it can be treated as a separate region. All trips to be performed on Texel are being outsourced to a third party in the new concession.

For this thesis, we will be concerned with the 'old' concession (disregarding Texel). Since we are interested in the historic demand of all trips for the past 2 years. We want to develop a schedule based on the historic demand using the new fleet. To evaluate the potential of a forecasting tool, we will compare the performance of the schedule based on historic demand - which can be used every day - with the well-tailored schedule for that day given a realization of demand for all trips. If it is beneficial using the forecasting model, it is useful to develop a model using this forecasting tool for the new concession.

Every week thousands of trips have to performed. Since there are no trips during the night and each vehicle returns to depot at the end of the day, the problem resets itself at night. Thus, 7 large MDVSP-instances have to be solved for each week.

3.2 Fleet

The new fleet of Connexxion consists of several types of buses. There are 3 **'Articulated'** buses with a length of 18 meters with a total capacity of 100 passengers. An articulated bus has 48 seats. There are 35 **'Standard'** buses with a length of 12 meters with each a capacity total capacity of 80 of which 42 seatings. The electric 12 meter bus BYD Midi, **'Electric Long'**, has a total capacity of 56, of which 26 are seats. For this type there are 19 buses present in the network. There are 54 VDL Midi **'Electric Small'** full-electric buses with a capacity of 19 passengers. Standing in these buses is prohibited. Lastly there are **'Taxi Buses'** with a capacity of 8 passengers which are deployed for special purposes.

Buses starting from a depot do have to return to that specific depot. It is possible for a bus to break down, which would result in that bus being returned to the maintenance station. In this thesis it is assumed that each depot has a fixed set of buses at the beginning of a workday. This is validated since there is a 'technical' reserve present for each type of bus at every single depot. In case of a break-down a bus from the technical reserve can be deployed. For the scheduling procedure however, we do not take break-downs during trips into account. In Table 1, the number of buses stationed at each depot is shown.

Table 1: Number of Buses per Depot

	Alkmaar	Hoorn	Den Helder
Electric Small	30	16	4
Electric Long	14	5	0
Standard	18	11	6
Articulated	0	0	3

It may be the case there are not enough vehicles available to drive the entire schedule. For example, there are more than 3 large buses required at the same time during the morning peak-hours. In such case, Connexxion can hire additional vehicles from a third-party for a fixed fee.

The costs for each trip is dependent of which type of bus is being deployed. The operational costs depend on the type of fuel used, the maintenance costs per vehicle type and lastly the anticipated damage costs. These costs vary during the lifetime of a bus. For convenience and without loss of generality, the average cost per kilometer is taken over the past 10 years. It is assumed that for all types of vehicles the driving-style for each bus-driver is homogeneous, such that the cost per kilometer is independent of the driver. Buses are linearly depreciated, independently of how many kilometers it has travelled during the year. Therefore, it is not included in either the cost per kilometer or the model itself.

An overview of the bus-fleet can be found in Table 2. Taxi Buses are not considered an option to deploy on trips. In this thesis, these buses are used for alternative transportation when customers are left behind.

Table 2: Overview Fleet Information

Type	Fuel	Seats	Capacity	Costs per KM
Articulated	Diesel	48	100	0.74
Standard	Diesel	42	80	0.43
Electric Long	Full-Electric	26	56	0.28
Electric Small	Full-Electric	19	19	0.18
Taxi Bus	Diesel	8	8	0.18

It is obvious that deploying small (electric) vehicles is beneficial as shown in Table 2. Though there are regulations for sufficient capacity to keep in mind. These regulations will be discussed hereafter.

3.3 Regulations for Capacity

A weekday is divided into separate distinct intervals (see Table 3). During each time-interval different rules are applicable for which type of bus can be deployed during a specific trip. During peak-hours for example, it is tolerated not being able to provide a seat to a customer. If a customer is required to stand in a bus, it is not permitted to let the customer stand for more than 30 minutes. Off peak hours, the company needs to provide a seat for each customer arriving. To determine which bus needs to be deployed the capacity of the bus is at least $\frac{5}{7}$ the monthly statistical occupancy rate of the busiest months. This regulation does not apply to trips frequented by students. For trips during peak-hours, the occupancy-rate may not exceed the number of seatings + 50 percent of places to stand.

Due to some reason it may happen that it is not possible to provide service to each customer at the stop. When it takes longer than 30 minutes for the next bus to arrive, Connexxion itself has to provide alternative transportation method on its own expenses. Otherwise, the customer may wait for the next bus to arrive. Besides the expenses of alternative transportation, a customer may file an official complaint to a governmental body. In case there are too many complaints, the governmental body may fine Connexxion.

In case of foreseeable increase in demand, such as public holidays or notifications of group-demand to the concession-holder earlier than 48 hours, the transportation company is required to deploy a larger or an additional vehicle.

Table 3: Overview Time Blocks

Weekdays	Saturdays	Sundays
Early Morning: Start Deployment - 7:00	Midday: Start Deployment - 18:00	Midday: Start Deployment - 18:00
Morning Peak-Hours: 7:00 - 9:00		
Midday: 9:00 - 16:00		
Afternoon Peak Hours: 16:00 - 18:00		
Early Evening: 18:00 - 21:00	Early Evening: 18:00 - 21:00	Early Evening: 18:00 - 21:00
Late Evening 21:00 - End Deployment	Late Evening: 21:00 - End Deployment	Late Evening: 21:00 - End Deployment

3.4 Restrictions for Electric Vehicles

Since diesel- and gas-buses are not limited by the action radius, they do not have to be refuelled during the day. Currently the battery, when fully charged, allows us to drive an electric bus for 250 kilometers before it needs to be recharged. The recharge-stations are located at every depot (Alkmaar, Den Helder and Hoorn). For every single bus there is a slow-charging recharge point. This way every single electric vehicle can start the day fully charged. Moreover, when a bus returns to its depot there is always a recharge point available.

3.4.1 *Assumptions*

The batteries used in electric vehicles are complicated. The chemistry used in the battery results in certain characteristics for recharging-speed, lifetime of the battery and efficiency of the battery in different seasons. To illustrate, recharging the battery from 0 to 80 percent takes about the same time as completely recharging the battery from 80 percent. The performance of a battery is season dependent. Especially in winter, when it is cold, the battery performs at about 70 to 80 percent efficiency.

In this thesis it is assumed that the recharge costs are incorporated in the costs per kilometer, such that the price per kWh is independent of the time of the day. For simplicity it is assumed that the lifetime of the battery does not depend on the depth of discharge, such that the costs are independent of the path a bus is deployed on. Moreover, it is assumed that for every single bus the battery has the same properties.

3.4.2 *Recharge-function*

Naturally the amount recharged is dependent of the time available to recharge and the current state of charge. If we were to assume a linear recharge-function the amount recharged would be independent of the current state of charge. To keep track of the state of charge would be a lot more simplified. However the recharge-function for each battery is far from linear. In Figure 3 three possible recharge-functions are shown. The black line is a representation of how a battery recharge-function could look like. The exact function is not known, thus we have to estimate the amount recharged. The blue line is the piece-wise linear approximation. Lastly the red line is a linear recharge function.

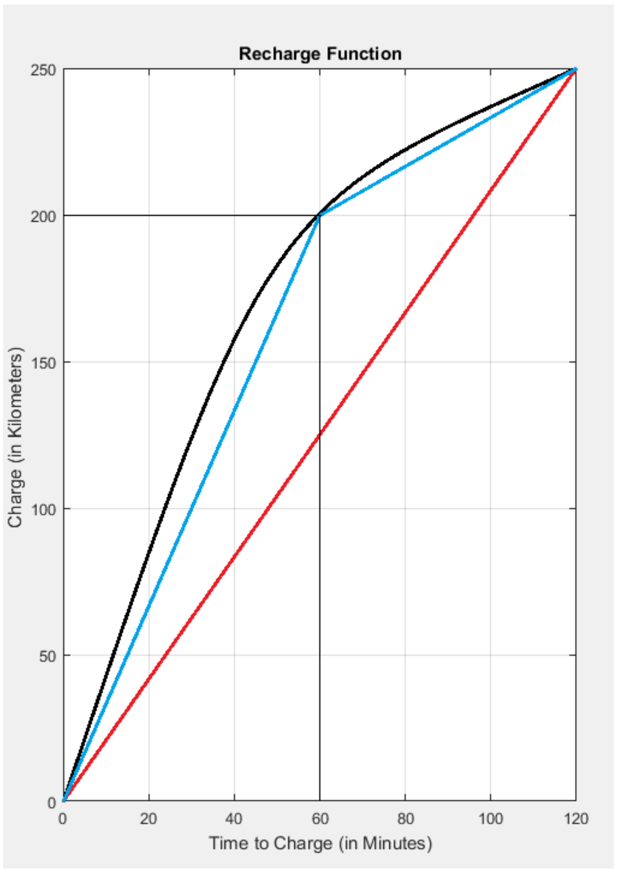


Figure 3: Recharge functions: State of Charge = 0

An important implication when assuming the recharge-function to be the red line, given that the battery is fully depleted, is that the amount recharged in one hour is severely underestimated. Similarly, given a state of charge of 200km, the red line would severely overestimate the amount recharged. The dangers of when the amount recharged is overestimated, is that we do allow for paths which in reality would result in a bus at the side of the road. Underestimating the amount recharged would result in fewer options to go to after a trip has been performed, limiting the number of feasible paths. The blue line (piecewise-linear estimation) will be used for the remainder of this thesis.

4 MATHEMATICAL FORMULATIONS

In this chapter we will elaborate on the notation used in the formulations. First we introduce the general notation for all sets, followed by a notation specifically used for this case. We will go into detail for recharge arcs and occupancy rates. Lastly we will introduce a set-covering formulation for which Column Generation will be applied.

4.1 General Notation

Denote the following sets: $\mathcal{D}$ the set of depots in the concession. Let $N = \{1, ..., n\}$ be the set of trips to be performed. Let $\mathcal{T}$ be the set of all vehicle types, each with a specific cost per kilometer kmc^k, number of seats s_k and a total capacity m_k. Let $\mathcal{K}$ be the set of all depots/vehicle combinations. The graph $G = (V, A)$ consists of a number of network-layers, each representing a sub-graph for a depot/vehicle combination. Since there are $|\mathcal{D}| \cdot |\mathcal{T}| = |\mathcal{K}|$ possible combinations, we need to make $|\mathcal{K}|$ network-layers. For each network-layer, there are r_k vehicles present. Each network-layer has its own specific set of nodes and arcs. Let V^k be the set of nodes corresponding to network-layer k. Let A^k be the set of all arcs in network-layer k.

The corresponding costs for each arc $(i, j) \in A^k$ depends on the type of vehicle and the distance travelled over arc (i,j), which is denoted by c^k_{ij}. There are no fixed costs available, since depreciation of buses is independent of whether it is being deployed or not. We define the cost for each arc (i,j) $\in A^k$: $c^k_{ij} = kmc^k \cdot (d_{ij} + td_j)$ where d_{ij} is the dead-head distance travelled from trip i to trip j and td_j is the distance travelled during trip j.

4.2 The Network

For each network-layer, the depot is represented by a source node (n+k) and a sink node (n+k'). Provided for each trip is a starting location l^s, end location l^e and starting time t^s and ending time t^e. Each trip is represented by a single node in every single network-layer. Therefore, the set of nodes for network-layer k: $V^k = \{(n+k) \cup N \cup (n+k')\}$.

The set of arcs in network-layer k A^k consists of arcs from the source node and trips, arcs between trips and arcs from trip to depot. A connection between trips can be made when $t^s_j \geq t^e_i$ and $l^s_j = l^e_i$ holds, or $t^s_j \geq t^e_i + \tau_{ij}$ where $l^s_i \neq l^e_j$.

After a bus finishes a trip, it has to go somewhere. It is either parked at the end station for some time, or it has to be returned to the depot. Most end stations have limited (if any) parking space, such that when a bus finishes a trip it quickly proceeds with a new trip or it should return to depot. We do not have information for parking space at stations, therefore let us assume that when a bus finishes a trip it should leave that station within 1 hour. In other words, it should continue with the next trip within 2 hours (allowing the bus to park 1 hour at both the end station of trip i and start station of trip j). Therefore, let trip j be compatible with trip i, $i \alpha j$, if and only if:

$$t_j^s \geq t_i^e + \tau_{ij} \tag{4.1}$$

$$t_j^s \leq t_i^e + \tau_{ij} + 120 \tag{4.2}$$

The resulting set of arcs A^k is the union of pull-out arcs, compatible trips and pull-in arcs:

$$A^k = \{(n+k, i) : i \in N\} \cup \{(i,j) \in N^2 : i \alpha j\} \cup \{(i, n+k) : i \in N\} \tag{4.3}$$

4.2.1 *Recharge Arcs*

To incorporate the action radius for electric vehicles, we need to keep track of the battery charge during the day. After a trip is finished, we should know whether an electric vehicle should return to depot to recharge or the vehicle can proceed with a new trip.

In order to create feasible paths for electric vehicles, we first need to identify opportunities to recharge. Whether there is a possibility to recharge is dependent of the time-to-recharge from trip i to trip j denoted by γ_{ij}. After trip i we would need to travel back to depot and proceed from depot to trip j. Therefore,

$$\gamma_{ij} = t_j^s - t_i^e - \tau_{i,Depot} - \tau_{Depot,j} \tag{4.4}$$

For simplicity, let us assume that when the time-to-recharge is more than half an hour ($\gamma_{ij} \geq 30$) it may be worthwhile to recharge. Recharging for i.e. 5 minutes is pointless as we need to cover additional distance to depot and it is harmful to the battery.

Given the opportunity to recharge, the network needs to accommodate for the decision to recharge or to proceed with a new trip. Therefore we introduce recharge arcs. A recharge arc exists when $\gamma_{ij} \geq 30$. Let us use the notation $i \beta j$ if trip j is recharge compatible with trip i. The set of arcs in network layer k becomes:

$$AR^k = \{(i,j) \in N^2 : i \beta j\} \tag{4.5}$$

Note that the possible connections we have removed from $\{(i,j) \in N^2 : i \alpha j\}$ are being replaced by the recharge arcs. For diesel buses in particular recharge arcs are simply depot arcs. Another note is that some trips may both be compatible and recharge-compatible with another trips.

4.2.2 *Occupancy rate*

For each trip a certain maximum occupancy rate (or demand) is given. Denote D_j the demand of trip j and Cap_j the appropriate capacity necessary for trip j. To incorporate the occupancy rate in the network, we can either program it as a hard constraint in our formulation or adjust the set of nodes such that vehicles with insufficient capacity cannot reach trips with high demand. The latter has the advantage of decreasing the number of nodes (and number of arcs) present in a network layer. To exclude those specific nodes from our network, we redefine the set of nodes:

$$V^k = \{(n+k)\} \cup \{j \in N : Cap_j \geq D_j\} \cup \{(n+k')\} \tag{4.6}$$

Given the regulations described in section 3.3, we need to consider whether we need capacity in terms of seats or seats plus standing places. For trips during peak hours, the occupancy rate may not exceed the number of seats + 50% of the standing places of the bus. During off-peak hours the appropriate capacity to use is the number of seats of the bus, since we need to be able to provide a seat to each customer. Therefore, the appropriate capacity for trip j used for trips is:

$$Cap_j = \begin{cases} s^k + \frac{1}{2}(m^k - s^k) & \text{if trip j is performed during peak hours} \\ s^k & \text{if trip j is performed during off-peak hours} \\ m^k & \text{if trip j is a trip frequented by students} \end{cases}$$

We have completed the network where we take both trip occupancy and recharging for electric vehicles into account. By removing arcs for which a bus has insufficient capacity, we ensure that in the final solution every passenger always has a seat in the off-peak hours and that no bus driver has to deny passengers at a bus stop.

4.3 Set-Covering Formulation

To conclude this chapter we will finish with the formulation.

We will use a similar formulation as described in section 2.3, where we define set $\mathcal{P}'$ the (restricted) set of feasible paths. Let the cost coefficient $c_p = \sum_{(i,j)\in A^k} c^k_{ij} x^k_{ij}$ the cost of a path p in network layer k. For each path in $\mathcal{P}'$, we define parameters $a_{ip} = 1$ if trip i is included in path p, 0 otherwise. For each network layer k there are r_k vehicles present. Last but not least we introduce decision variable $\theta_p = 1$ if path p is used, 0 otherwise.

We also have to introduce an additional feature: Given the number of buses in each depot, there may not be a feasible solution at all due to a shortage of buses. Connexxion still has an option to rent buses at a third-party, such that each trip can be covered. To incorporate the decision variable of renting additional vehicles, we denote y_k the number of rented vehicles for network layer k. Denote c^{rent}_k the costs of renting an additional vehicle in network layer k. The resulting formulation is stated below:

$$\begin{aligned}
& min \sum_{k \in K} \sum_{p \in \mathcal{P}^k} c_p \theta_p + \sum_{k \in \mathcal{K}} c_k^{rent} y_k && && (4.7)\\
& s.t. \sum_{k \in K} \sum_{p \in \mathcal{P}^k} a_{ip} \theta_p \geq 1 && \forall i \in N && (4.8)\\
& \sum_{p \in \mathcal{P}^k} \theta_p \leq r_k + y_k && \forall k \in \mathcal{K} && (4.9)\\
& \theta_p \in \mathbb{B} && \forall p \in \mathcal{P}' && (4.10)\\
& y_k \in \mathbb{N} && \forall k \in \mathcal{K} && (4.11)
\end{aligned}$$

The objective function is the sum of the operational costs of using paths and the rental costs. The two main constraints are that every single trip has to be performed at least once (4.8) and that we cannot exceed the amount of vehicles that we have at our disposal (4.9). In order to get a proper solution, the solution has to be integer. Constraint sets (4.10) and (4.11) are the integrality constraints.

In this formulation we require for each trip to be covered **at least** once. For the column generation approach (or any heuristic in general) to find a solution where each trip is covered exactly once takes more time.

We do not have to incorporate both the occupancy rate and action radius for electric vehicles in this formulation. First, the occupancy rate has been included in the network, such that sufficient capacity is being deployed for every trip. Secondly, the paths we choose are feasible in terms of action radius.

5 SOLUTION APPROACH

In this chapter we will discuss the heuristic we have constructed. In short, we have constructed a Truncated Column Generation heuristic to solve the VSP as described in the previous chapters. We have proposed a formulation for the Pricing Problem which we solve heuristically. We continue elaborating the heuristic used to solve the pricing problem for both the conventional diesel buses and electric vehicles.

This chapter will start with the Restricted Master Problem and its LP-relaxation. Secondly, we will discuss the corresponding pricing-problem for both conventional diesel buses and electric vehicles. We will elaborate on the dedicated heuristic which is used to solve the Pricing Problem. We conclude this chapter with the description of the overall heuristic.

5.1 Restricted Master Problem

For the Restricted Master Problem we use the formulation (4.7)-(4.11) from the previous chapter. In order to apply Column Generation, we need to solve the **LP-Relaxation**:

$$\min \sum_{k\in K}\sum_{p\in\mathcal{P}^k} c_p\theta_p + \sum_{k\in\mathcal{K}} c_k^{rent} y_k \tag{5.1}$$

$$s.t. \sum_{k\in K}\sum_{p\in\mathcal{P}^k} a_{ip}\theta_p \geq 1 \qquad \forall i \in N \tag{5.2}$$

$$\sum_{p\in\mathcal{P}^k} \theta_p \leq r_k + y_k \qquad \forall k \in \mathcal{K} \tag{5.3}$$

$$0 \leq \theta_p \leq 1 \qquad \forall p \in \mathcal{P}' \tag{5.4}$$

$$y_k \geq 0 \qquad \forall k \in \mathcal{K} \tag{5.5}$$

Introducing the dual variables μ_j for each cover constraint in constraint set (5.2) and dual variables λ_k for each vehicle constraint in constraint set (5.3). We use this formulation to get the LP lower-bound. We are interested in this value as it indicates the quality of the integer solution.

5.2 The Pricing Problem

Given the dual variables of the RMP, the objective is to find new paths with negative reduce costs. λ_k and μ_j are the dual variables resulting from the Restricted Master Problem. Let x_{ij}^k be the decision variable for arc (i,j) in network layer k. Note that we are solving the pricing problem for every depot/vehicle type combination. Let parameter m_{ij} be the amount of charge used (or gained in case of a recharge arc) on arc (i,j) and let R^k be the action radius for vehicles in network layer k. For each network layer k the pricing problem can be formulated as follows:

$$SP_k : min - \lambda_k + \sum_{(i,j)\in A^k} (c_{ij} - \mu_j)x_{ij}^k \tag{5.6}$$

$$s.t. \sum_{i\in V^k} x_{ij}^k - \sum_{i\in V^k} x_{ji}^k = 0 \qquad \forall j \in N \tag{5.7}$$

$$\sum_{i\in N} x_{n+k,i}^k = 1 \tag{5.8}$$

$$\sum_{i\in N} x_{i,n+k}^k = 1 \tag{5.9}$$

$$\sum_{i:(i,j)\in A^k} m_{ij} \cdot x_{ij}^k \leq R^k \qquad \forall j \in N \tag{5.10}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in V^k \tag{5.11}$$

We define this formulation as the Fuel Constrained Shortest Path Problem. The objective function (5.6) is the length of the path. Constraints (5.7) are the 'continuation' constraints: when a trip is finished, the bus will proceed from this node. The vehicle starts from the depot and it ends at the depot where it has started from by using constraints (5.8) and (5.9).

For any type of vehicle we cannot exceed its action radius. Constraint set (5.10) implies this range constraint. However, by adding this set of constraints the pricing problem becomes a lot more complicated. Therefore, we need a heuristic which can take the action radius into account. In the next section we elaborate on a dedicated heuristic incorporating the action radius of electric vehicles in order to find a feasible path.

For diesel vehicles constraint set (5.10) is redundant. Therefore, for diesel vehicles this formulation boils down to a regular Shortest Path Problem. By incorporating the action radius of vehicles into an algorithm, we will be able to use the same algorithm for diesel vehicles and electric vehicles as well.

5.3 Dedicated Algorithm for Pricing Problem

The Pricing Problem can be solved using this IP-formulation, though due to the complexity of the problem we have chosen to solve it heuristically. For any Shortest Path Problem algorithms like Dijkstra or the Bellman-Ford algorithm can be used. However, given the additional constraint of a limited action radius, we have to develop a dedicated heuristic.

Network

First notice that each network-layer is a directed acyclic graph (DAG). Intuitively, we cannot find a path which leaves a node and eventually return to the same node. This would mean a bus is deployed on a trip and a successive trip is scheduled earlier that day. We cannot travel back in time. Using this property, we can sort the nodes in each network layer such that all edges are pointed in the same direction. To visualize this new representation of the network, a topological sorted network is shown in Figure 4.

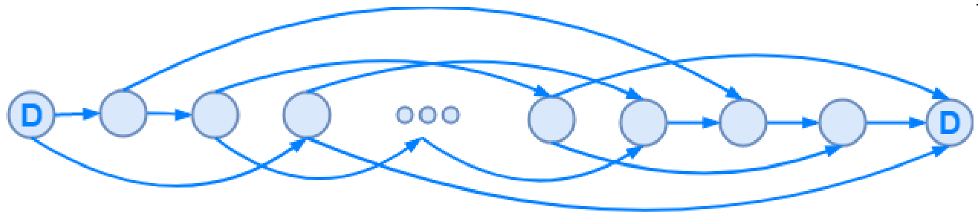


Figure 4: Topological Sorting

This allows us to search for any shortest path through the network more efficiently. By representing the network in this manner, we can work from left to right through the network. In each iteration the shortest path for the next node is found. This algorithm results in a shortest path from depot to depot.

Main Idea

Similar to the Bellman-Ford and Dijkstra algorithm, the main idea of the algorithm is to update the shortest paths to nodes only when it results in a new shortest path. In this case we want to incorporate the additional constraint. To do so, we only update the shortest path when there is a better result **and** there is still sufficient battery left to return to the depot. We keep track of the remaining charge left in battery and update the shortest paths only when there is sufficient charge left. When there is insufficient charge, the vehicle cannot proceed with a new trip. Instead, it has to return to the depot to recharge.

Update Criteria

For each arc between the current node v and the neighbouring node u, we check whether a new shortest path to node u has been found and whether it is feasible in terms of charge using the following requirements:

$$Dist[u] > Dist[v] + c^k_{vu} \tag{5.12}$$

and

$$SoC[v] - td_u - d_{uv} - d_{u,depot} > 0 \tag{5.13}$$

We define $Dist[u]$ as the current length of the shortest path to node u and c^k_{vu} the cost of the arc (v,u) in network layer k. Let $SoC[v]$ be the remaining charge in battery at node v, td_u the distance covered during trip u and d_{uv} the deadhead-distance from node u to node v.

Equation (5.12) states that the new path to node u is the shortest path in terms of reduced costs. For the new shortest path to node u to be feasible, the state of charge at node v needs to exceed the trip distance of node u and dead-head distance of arc (v,u) travelled **and** still have enough charge left to reach the depot before running out of charge. As stated in equation (5.13). Note that for diesel vehicles equation (5.13) is always satisfied as we assumed diesel vehicles can be used the entire day without refuelling.

When using a recharge arc, we need to update the current state of charge according to the recharge function (Appendix B) resulting in the updated state of charge SoC^+. The requirements become:

$$Dist[u] > Dist[v] + c^k_{v,depot} + c^k_{depot,u} \tag{5.14}$$

and

$$SoC^+ - td_u - d_{u,depot} > 0 \tag{5.15}$$

For equation (5.14), we have to consider both routes the bus needs to drive. First, the vehicle has to go from the end-station of the last trip to the depot and from the depot to the start-station of the successive trip. Equation (5.15) is similar to the corresponding equation for regular arcs. We still have to consider the two routes which are being covered: for node v to the depot and from the depot to node u. However, we do not explicitly state the distance travelled from node v to the depot. This is already included in the calculations of SoC^+.

Overview Algorithm

An overview of the algorithm is given in Algorithm 1. We initialize the algorithm with the path distances to each node equal to ∞. Where the predecessor of node v is undefined. To keep track of the state of charge we denote $SoC[v]$ the state of charge after performing trip v. Lastly, we introduce $Recharge[v]$, which denotes whether we are using a recharge arc to node v or not.

Introducing two depot nodes results in a total of $|N| + 2$ nodes in the network. Thus we need to perform $|N| + 2$ iterations per sub-problem to find the shortest path in that network-layer. For each iteration we get the next node in the topological order and determine the set of neighbouring nodes through compatible arcs and recharge compatible arcs.

Per iteration we determine for the current node the set of compatible arcs and the set of recharge compatible arcs. We determine for each arc in these two sets whether it results in a new shortest path to node u using the update-criteria as discussed before.

Finally, in order to retrieve the shortest path for network G^k, we need to back-track using $Pred[v]$ and $Recharge[v]$. Starting with the ending depot node, we obtain the predecessor and add this node to the list. When a recharge arc has been used, the depot node is also added to the list. We continue this process until we end up with the starting depot node.

The complexity of this algorithm is $\mathcal{O}(|A^k|)$ since we are only evaluating the arcs. Essentially it runs in linear time considering the number of arcs (which is quadratically related to the number of nodes present). So overall, we have an algorithm which performs quite efficiently. In section 5.4 we will go into detail whether the path found is indeed the shortest path in the network.

Algorithm 1: Shortest Path Algorithm

```
Data: Topological Ordering of $G^k : L^k$
Action radius $R$ of the bus type in network layer k
Result: Shortest Path $\hat{p}$
Initialization:
foreach $v \in V^k$ do
    Set distance to node v: $Dist[v] \to \infty$
    Set predecessor for node v: $Pred[v] = \emptyset$
    Set State of Charge at node: $SoC[v] = R$
    Set Recharge at node: $Recharge[v] = 0$
end
Set distance source node: $Dist[s] = 0$
Set count = 0;
while $count < |N| + 2$ do
    Get next node v in the topological order
    Determine the set of all compatible trips for trip v := $\mathcal{S}(v)$
    Determine the set of all recharge compatible trips for trip v
     := $\mathcal{R}(v)$ foreach $u \in \mathcal{R}(v)$ do
        Update State of Charge: $SoC^+$;
        if $Dist[u] > Dist[v] + c^k_{v,depot} + c^k_{depot,u}$ and
         $SoC^+ - d_u - d_{depot,u} - d_{u,depot} > 0$ then
            $Dist[u] = Dist[v] + c^k_{v,depot} + c^k_{depot,u}$
            $Pred[u] = v$
            $SoC[u] = SoC^+ - d_{depot,u} - d_u$
            $Recharge[u] = 1$
        end
    end
    foreach $u \in \mathcal{S}(v)$ do
        if $Dist[u] > Dist[v] + c^k_{vu}$ and
         $SoC[v] - d_u - d_{vu} - d_{u,depot} > 0$ then
            $Dist[u] = Dist[v] + c^k_{uv}$
            $Pred[u] = v$
            $SoC[u] = SoC[v] - d_u - d_{vu}$
            $Recharge[u] = 0$
        end
    end
end
Back-tracking:
value = depot
while $value \neq StartDepot$ do
    add value to $\hat{p}$
    if Recharge[value] == 1 then
        add depot to $\hat{p}$
    end
    value = Pred[value];
end
add depot to $\hat{p}$
Return: $\hat{p}$
```

5.4 Proof of Correctness

Proposition 1: The algorithm to solve the pricing problem, as described above, results in an optimal solution for diesel buses.

We are proving this statement using Dynamic Programming and the topological sorting of the network-layer. We can show that this method indeed results in the shortest path, at least for diesel fuelled vehicles. For dynamic programming, denote $f(k)$ the cost of the shortest path from the source node to node k in the topological sorting and initialize with:

$$f(s) = 0 \tag{5.16}$$

$$f(k) = \infty \qquad \forall\, k \in N/s \tag{5.17}$$

Where s is the source (depot) node. Naturally, we start with the current cost equal to 0 to the source node. We define the current cost to every other node in the network equal to ∞. The objective is to find the shortest path from the depot node s to depot node t. For each iteration in the Dynamic Programming algorithm we are using the induction step:

$$f(k) = min_{i \in \{s,...,k-1\}}\{f(i) + c(i,k)\} \tag{5.18}$$

Does the induction step result in the shortest path to node k? Given the topological sorting, the first iteration results in the shortest path to the first scheduled trip in the schedule. Since the only incoming arc for trip 1 is coming from the depot, we know for sure that $f(1) = f(0) + c(0,1) = c(0,1)$ is the minimum cost for reaching trip 1.

So if we were to assume after k iterations we know all shortest path to nodes $\{s, 1, ..., k\}$ where $f(i)$ is the cost of the shortest path to node i in the topological sorting. The shortest path to node (k+1) is found through all incoming arcs from node i to node (k+1). Therefore, the cost of shortest path for node (k+1) is denoted by:

$$f(k+1) = min_{i \in \{s,1,...,k\}}\{f(i) + c(i,k+1)\} \tag{5.19}$$

Since the path found using dynamic programming is always feasible for diesel vehicles, we know by mathematical induction it results in a shortest path for diesel vehicles.

We have shown the algorithm does provide an optimal solution for diesel vehicles. However, is this statement also valid for electric vehicles? It is likely that an electric vehicle will run out of charge when deployed on that specific path. To prevent any unwanted stops at the side of road, we have introduced an additional criterion such that there will always be sufficient charge. But will the algorithm still return the optimal path?

Proposition 2: The algorithm to solve the pricing problem for electric vehicles does not result in an optimal solution.

Let us consider the following example as shown in Figure 5. We have three trips: trip i, j and k. From trip i we can either use a recharge arc (the dotted arc) or a regular arc to get to trip j, and from trip j we can only use a regular arc to get to trip k.

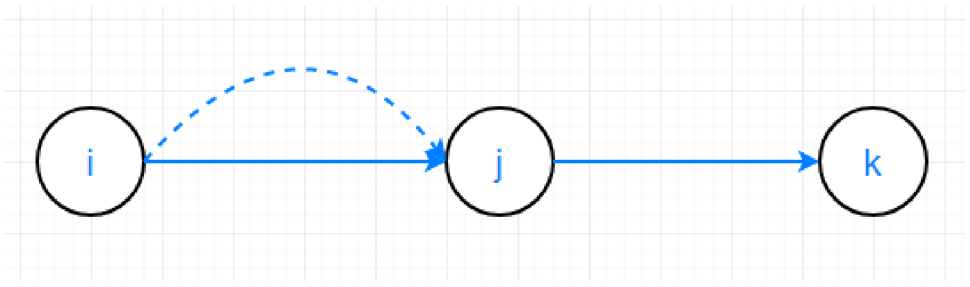


Figure 5: Recharge arc example

Let us assume that there is enough charge left in the battery such that both trip i and j can be completed when we are using the regular arc, but not enough in that case to also continue with trip k. If we were to choose the recharge arc instead, we do have enough charge to complete all three trips in this example.

In such a case, the algorithm fails in terms of optimality. Since the regular arc is the cheapest to use, the algorithm will use this arc to reach trip j. However, it will not evaluate the recharge arc one step further. Therefore, the algorithm is a bit short-sighted. Next, we will provide a scenario where it goes wrong:

Let the current shortest path to node k be through node j. Such that $f(k) = f(j) + c(j,k)$. Node k cannot be reached through node i due to the remaining charge left in battery, as visualized in Figure 6. Notice the differences in arcs for trips i,j and k compared to Figure 5.

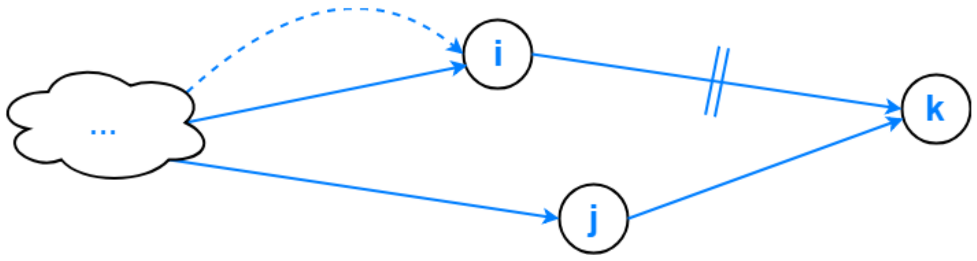


Figure 6: Example Pathing Electric Vehicle

Now consider the following: when we are using a recharge arc to node i, which results in a sub-optimal path with cost $f(i)' > f(i)$, but allows us to travel through node i to node k. Is it possible that for any combination of nodes i, j and k that $f(k) = f(j) + c(j,k) > f(i)' + c(i,k)$ - such that we are using a sub-optimal path to find a 'better' shortest path?

Provided that we are dealing with large (in absolute terms) negative dual variables that are incorporated in the cost of each arc, it is indeed possible that a sub-optimal path may lead to a shortest path. Especially in the first iterations of the heuristic, where the reduced costs for each trip equals $-M$. Therefore, the difference between $f(i)$ and $f(i)'$ is negligible, while the difference between $f(j)$ and $f(i)'$ may be larger than M. Resulting in the 'sub-optimal' path through node i being better than the 'shortest path' through node j. Therefore, we conclude the resulting path for electric vehicles is not optimal.

5.5 Truncated Column Generation Heuristic

The algorithm is similar to the truncated column generation algorithm described in (Pepin et al. 2009). Similar to any column generation heuristic, a starting solution Λ is introduced. This starting solution is a feasible solution which covers each trip at the cost of big M. The only purpose of this solution is to provide a starting point for the heuristic. The next step is solving the RMP, followed by solving the subproblems which results in new paths.

One of the inevitable disadvantages of any column generation heuristic is the tail-off effect. The improvements in objective values become smaller and smaller, but still keeps improving nonetheless. In order to speed up the process and, more importantly, achieve an integer solution, we need to interfere during the column generation process. This will, however, result in the fact that we will not achieve the optimal solution.

One important deviation from (Pepin et al. 2009) is the early termination criterion. The paper states an early termination criterion of a minimum number of iterations without exceeding the minimum **absolute** decrease. In our algorithm we will be using a **relative** decrease of the objective value compared to the previous iteration. We think that the relative decrease is much more informative than absolute decrease. When the improvement of the next iteration is less than $Z_{MIN}-\%$ compared to the previous solution, we are fixing variables.

When the early termination criterion is met, such that the iteration does not result in sufficient decrease, we are going to fix variables. For each variable θ_p exceeding θ_{MIN} we set the lower-bound of this variable to 1. If no such variable exists, we set the variable with the highest value to 1. One possible interpretation of θ_{MIN} is the minimum potential of a path. In other words, the relevant path would at least be used in $(\theta_{MIN} \cdot 100)\%$ of all solutions.

The heuristic may provide us a solution where trips are performed multiple times. To remove these duplicate trips, we are interested in which path is assigned to the smallest bus. Using this path and removing all others will result in the cheapest solution. When there are still multiple paths left, for example three paths are driven by the standard diesel bus, we keep the trips in the first path and eliminate the remaining duplicates. An overview of the process is described in Algorithm 2.

Algorithm 2: Truncated Column Generation Heuristic

```
Data: Network
Result: Integer Solution
Step 0: Initialization
    Starting solution $\Lambda$
    Choose parameters $Z_{min}$, $\theta_{min}$
Step 1: Solve the RMP
    Denote primal solution $(\theta_n, \Lambda_n)$ and dual solution $(\mu_n, \lambda_n)$.
    Define objective value $Z_n^{RMP}$
    $n \leftarrow n + 1$
Step 2: Early Termination Test
if $\frac{Z_{n-1}^{RMP} - Z_n^{RMP}}{Z_{n-1}^{RMP}} \cdot 100 < Z_{min}$ then
  | Go to Step 5
end
Step 3: Solve Sub-Problems
foreach Sub-problem $SP_k : k \in K$ do
  | Update the cost of arcs in $A^k$ given the dual-variables
  | Solve the Shortest Path Problem on $G^K$
  | Denote solution values $Z_n^{SP,k}$
end
Step 4: Update Restricted Master Problem
    Given any shortest paths with reduced costs: $Z_n^{SP,k} < 0$
    Update the set of paths $\mathcal{P}'$
    Go to step 1.
Step 5: Feasibility Test
if $\Lambda_n \neq 0$ then
  | Stop: No feasible solution has been found
end
Step 6: Integrability Test
if Solution is integer then
  | A solution has been found!
  | Go to Step 8
end
Step 7: Rounding
foreach $p \in \mathcal{P}'$ do
  | if $\theta_p > \theta_{min}$ then
  |   | $\theta_p \leftarrow 1$
  | end
  | if no such variable exists then
  |   | set $\theta_p^* \leftarrow 1$, where $\theta_p^*$ is the maximum value found.
  | end
end
    Go to Step 1
Step 8: Remove duplicate trips
```

6 DATA

In this chapter we will elaborate on the data itself and the alterations being made. Next, we need to determine which bus can be deployed on a each trip in the timetable. To do so, the distribution of the occupancy per trip is derived and a confidence value can be calculated. Lastly, we introduce two test-weeks for real-life instances. The data set includes all scheduled trips from 1 January 2016 to 31 March 2018.

6.1 Alterations to the data-set

In Appendix A the buslines for each region within the concession is shown. There are six types of buslines:

- The regular lines: 001-399
- The neighbourhood-lines: 401-499
- The student-lines: 601-699
- On-demand-lines: 701-799
- Beach-lines: 801-899
- Night-buses: N60 and N69

For the On-demand buslines (buslines 701-799) every registered trip in the data-set has been flagged as 'cancelled'.

Since there has not been a trip for any of these bus-lines there is no reason to keep them in the data set. Therefore these trips have been removed.

Trips performed in the middle of the night, or night-buses, are only scheduled in weekends (Friday night and Saturday night) and are only available between Alkmaar - Heerhugowaard and Alkmaar - Amsterdam. On Friday night it is only bus-line N60: Alkmaar - Heerhugowaard, for which only 7 trips have to be driven. On Saturday night 15 trips have to be performed. Since this part of the solution is trivial, it has been excluded from the data set. Neighbourhood-lines, student-lines and beach-lines will be treated like regular lines, with the regulations described in Section 3.3 still applicable.

6.2 Occupancy Rate

In Table 4 an example of available data is shown. This example is an overview of all the trips made on a specific date for a specific busline in a certain direction. We are interested in how many passengers were on the bus at the busiest moments of a specific trip. Data of the occupancy rate is known for each trip for every single day for the past 2 years. The customer demand for a trip is non-fractional. Therefore, the occupancy rate follows a discrete distribution.

Table 4: Data Example Busline 1

Date	Busline	Direction	Trip	Starting time	End time	Occupancy	Start Location	End Location
1-2-2018	M001	2	2	06:33:00	06:50:00	6	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	4	07:33:00	07:50:00	5	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	6	08:33:00	08:50:00	4	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	8	09:23:00	09:50:00	7	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	10	10:23:00	10:50:00	2	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	12	11:23:00	11:50:00	5	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	14	12:23:00	12:50:00	6	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	16	13:23:00	13:50:00	5	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	18	14:23:00	14:50:00	9	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	20	15:23:00	15:50:00	10	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	22	16:23:00	16:50:00	12	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	24	17:23:00	17:50:00	6	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	26	18:23:00	18:47:00	4	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	28	19:23:00	19:41:00	5	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	30	20:23:00	20:41:00	5	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	32	21:23:00	21:41:00	3	Alkmaar Station	Alkmaar Station
1-2-2018	M001	2	34	22:23:00	22:41:00	3	Alkmaar Station	Alkmaar Station

In order to derive a distribution for the occupancy rate for each trip, we first need to sort by 'comparable' trips. Which means, trips which are performed on the same weekday, with the same starting- and ending location and the same starting time and ending time. Throughout the period from 1-1-2016 to 31-3-2018 various minor changes have been made to the schedule. Those changes typically are trips starting a few minutes earlier or later than before, generally to ensure a customer can catch a train or a different bus.

To bypass this problem, we can pick a week in 2017, since there are sufficient 'comparable' trips for each trip in the given timetable. Given these comparable trips, a distribution can be derived for the maximum occupancy rate of a specific trip.

6.3 Fitting Distributions

Let's discuss one trip that is performed every weekday. The start location is Heerhugowaard Crematorium starting at 07:50:00 and ends in Alkmaar Station at 08:21:00. For this trip a bar plot of comparable trips is shown and a histogram based on the occupancy rates. Using this histogram, we can make a guess of a distribution which fits the shape best.

Since the occupancy rate for trips follow a discrete distribution we will estimate the parameters for the uniform distribution, negative binomial distribution, the geometric distribution and poisson distribution. Based on the Akaike Information Criterion (AIC) we will decide which distribution fits best. In Table 5 the results are shown for this particular trip.

Table 5: Fitting Distributions

Distribution	Parameter 1	Parameter 2	Log-Likelihood	AIC
Uniform	66.0		-276.5	555.0
Negative Binomial	5.792	0.1296	-284.3	572.7
Geometric	0.0251		-313.1	628.3
Poisson	38.90		-401.5	805.0

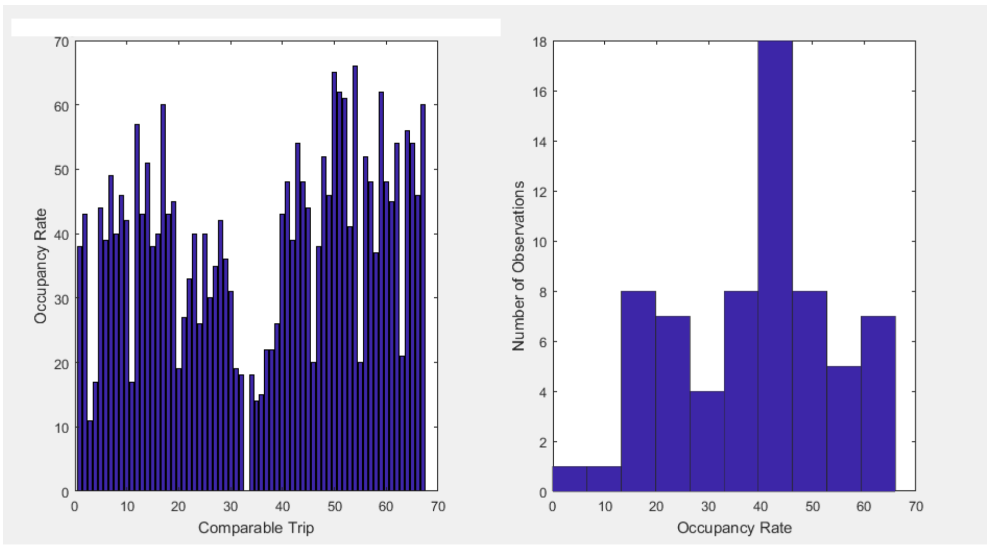


Figure 7: Busline 4: Heerhugowaard Crematorium - Alkmaar Station : 07:50:00 - 08:21:00

Based on this table we can conclude the uniform distribution fits 'best' for the occupancy rate of comparable trips. The parameter provided from this table is the maximum value of occupancy rates of the comparable trips. For negative binomial the parameters $[r, p]$ are estimated, where r is the number of successes and p the probability of success. For the geometric distribution the method returns an estimated value $\hat{p}$ for p, the probability for success. Lastly, the poisson-distribution parameter λ is estimated.

Given the fitted distributions we can easily determine the value of occupancy rate for which we, when there is sufficient capacity deployed given that occupancy rate, we have sufficient capacity with at least 95% confidence.

$$\begin{aligned} P[N \leq n] &\geq 0.95 \\ &= \sum_{k=0}^{n} \frac{1}{a-b} \\ &= \frac{n}{66} = 0.95 \\ &\rightarrow n = 63 \end{aligned}$$

Provided that this trip is performed during peak hours, the bus deployed for this trip does not necessarily need to provide a seat to each passenger, therefore a bus with total capacity exceeding n is allowed. Therefore, in the yearly schedule, this trip should be driven by a standard diesel bus. However, for many day-specific instances, where the trip occupancy is lower than n, a large electric vehicle could be deployed.

6.4 Test Weeks

Since the number of trips to be performed in a week is inconsistent over time, we chose a week starting in September 2017 and a week starting in January 2017 to use as a test week: Sunday 17 September 2017 - Saturday 23 September 2017 and Sunday 29 January - Saturday 5 February. After the alterations discussed above, there are 2241 trips to be performed on weekdays, 1595 trips on Saturday and 963 trips on Sunday.

These two test weeks result in 10 instances of the problem, excluding the weekends. We will use these instances to compare the solutions using different values of the minimum relative decrease parameter Z_{MIN}, such that we can say something more about the trade-off in quality of the final solution and the time necessary for the heuristic to solve these instances. The two test weeks also allow us to compare the current method of solving the instances using historical data, with the new proposed method of solving the vehicle scheduling problem on a daily basis.

7 RESULTS & DISCUSSION

In this chapter we report the results found using the method described in the previous chapter. Starting with the problem as described in chapter 3, the problem will be solved using the heuristic we have introduced in chapter 5. The trip occupancy we are using are determined as described in chapter 6.

For this thesis we are interested in the performance of the heuristic and the developments during the process. In order to find an answer of how well (or bad) the heuristic is performing, the final result will be compared with the approximation of the LP lower bound.

For the case study itself we are particularly interested in the influence of parameter Z_{MIN} on the time required for the heuristic to find a solution and the quality of the solution. Especially since there is limited time available to solve a day-specific problem. The second point of interest is whether additional vehicles are needed to solve the problems.

7.1 Computer Specifications

We are solving the problems using two different sets of computers. The first being a personal computer, equipped with an Intel Core i7 4510U processor clocked at 2.00 GHz and 8 GB RAM. The second being the computers available at the university, equipped with an AMD A4 PRO-7300B APU clocked at 3.80 GHz and 16 GB RAM. The first computer is used for solving the problems for schedules based on historical demand and the day-specific problems are solved using the computers available at the university.

To put this into perspective: the personal computer took 29997 seconds to solve the robust problem, while a computer on campus took 48219 seconds to solve the same problem which is an increase of 60.7% compared to my PC. As a rule-of-thumb, the configuration of the personal computer is a factor 1.6 faster than the computers on campus. To provide a clear overview of results, we divide the computation times for problems solved by the computers on campus by 1.6.

7.2 Initialization

We initialize the heuristic with the following parameters:

- $Z_{MIN} = 0.01$
- $\theta_{MIN} = 0.70$
- $c^{rent} = 500$

By setting $Z_{MIN} = 0.01$, we will stop the column generation process for a moment when the relative decrease comparing to the previous iteration becomes smaller than 0.01%. We will evaluate our choice of

Z_{MIN} in the following sections and provide a sensitivity analysis for different values of Z_{MIN}.

The choice of θ_{MIN} is arbitrary. Though, in literature $\theta_{MIN} = 0.70$ seems to be the norm.

Setting $c^{rent} = 500$ implies that we are using the buses that are still at the depot first. We will only rent additional buses when it is absolutely necessary as c^{rent} is more expensive than any path resulting from the sub-problems. Of course, in reality the costs of renting a bus for a day may be lower (or higher). However, we prefer to use every single bus in the initial bus-fleet before we start renting additional buses.

7.3 Schedules Based on Historic Demand

The first problem we will solve using the heuristic is the 'robust' schedule, which can be deployed every day. We have determined the 95% confidence values for the trip occupancy for every single trip using the observations of all weekdays. Thus, the resulting schedule can be deployed each weekday for the duration of the concession.

We will focus on the performance of the heuristic, as we are particularly interested in how the solution develops over time, given the parameters used in the initialization. We solve the LP-relaxation of the RMP to get a theoretical lower bound. Any column generation heuristic is notorious for the tail-off effect. Therefore, we set a time limit of 10 hours for the LP-relaxation to solve.

Since our algorithm to solve the pricing problem does not find the optimal solution for electric vehicles, we only find an approximation of the LP-lower bound. Besides, setting a time limit for solving time results in an overestimation of the lower bound.

7.3.1 *Heuristic Performance*

In Figure 8 the progression of the objective value is shown for both the heuristic and the LP-relaxation. For every iteration the current objective value is shown - only after the final iteration of the heuristic do we have a feasible integer solution. Due to the initial solution, the objective value after the first few iterations is very large. The tail-off effect is clear to see when the heuristic is running for some time: The heuristic is still improving the objective value, although the improvement is minimal. Therefore, we suggested a minimal **relative** decrease. When the improvement is smaller than $Z_{MIN} - \%$ compared to the previous iteration, the heuristic will fix variables and continue the process.

In Figure 9 the objective value of the final 1000 iterations are shown. This is especially interesting, as the heuristic is 'rapidly' fixing variables. After every few iterations, a peak in objective value is shown. These peaks naturally occur after a variable has been fixed. In the last few iterations we see the peaks being larger and occurring in a more rapid succession than previous peaks.

At the end of the process - when the integer solution is almost complete as most paths have been chosen - we have that most trips are

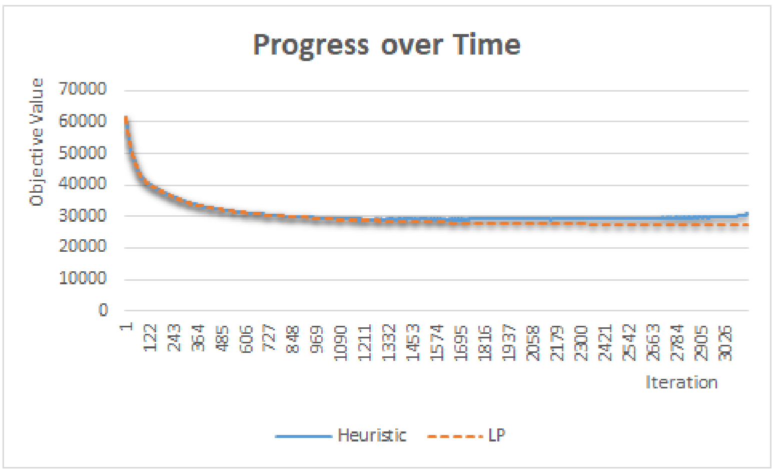


Figure 8: Heuristic - LP progress over time

driven at least once. Therefore, the improvement that can be achieved through the last trips is very much limited, which results in faster variable fixing. Since the renting costs of an additional bus are sufficiently high, the heuristic only hires those when absolutely needed at the end of the process. This explains the larger peaks at the end.

Using the value of $\theta_{MIN} = 0.70$, we see during the process only one variable at the time is being fixed. This result is an indication for the problem being highly degenerate. The variable (path) being fixed has the highest value, though it may not differ that much compared to other viable paths.

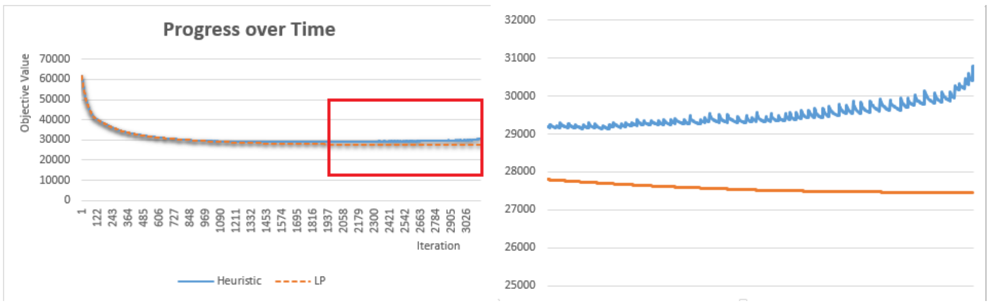


Figure 9: Progress over Time - Final Iterations

Given the initialization, the heuristic takes 29997 seconds (which is about 8 hours and 20 minutes) to find an integer solution with objective value equal to 30471. During this process, the heuristic spends the most time solving the RMP: 29386 seconds. About 2 percent of the time is spent solving the sub-problems. We know that 606 seconds were spend solving the sub-problems and more than 3000 iterations were needed to get an integer solution. During each iteration for each vehicle/depot combination a shortest path is found, therefore during each 12 sub-problems are being solved. Thus, each sub-problem takes about 0.01 seconds to solve. After 12 hours we find the approximation of the lower bound $z^{LP} = 27507$. Therefore, we conclude that the gap between the objective value of the integer solution and the approximated LP-lower bound is 10.6%.

7.3.2 *The Solution*

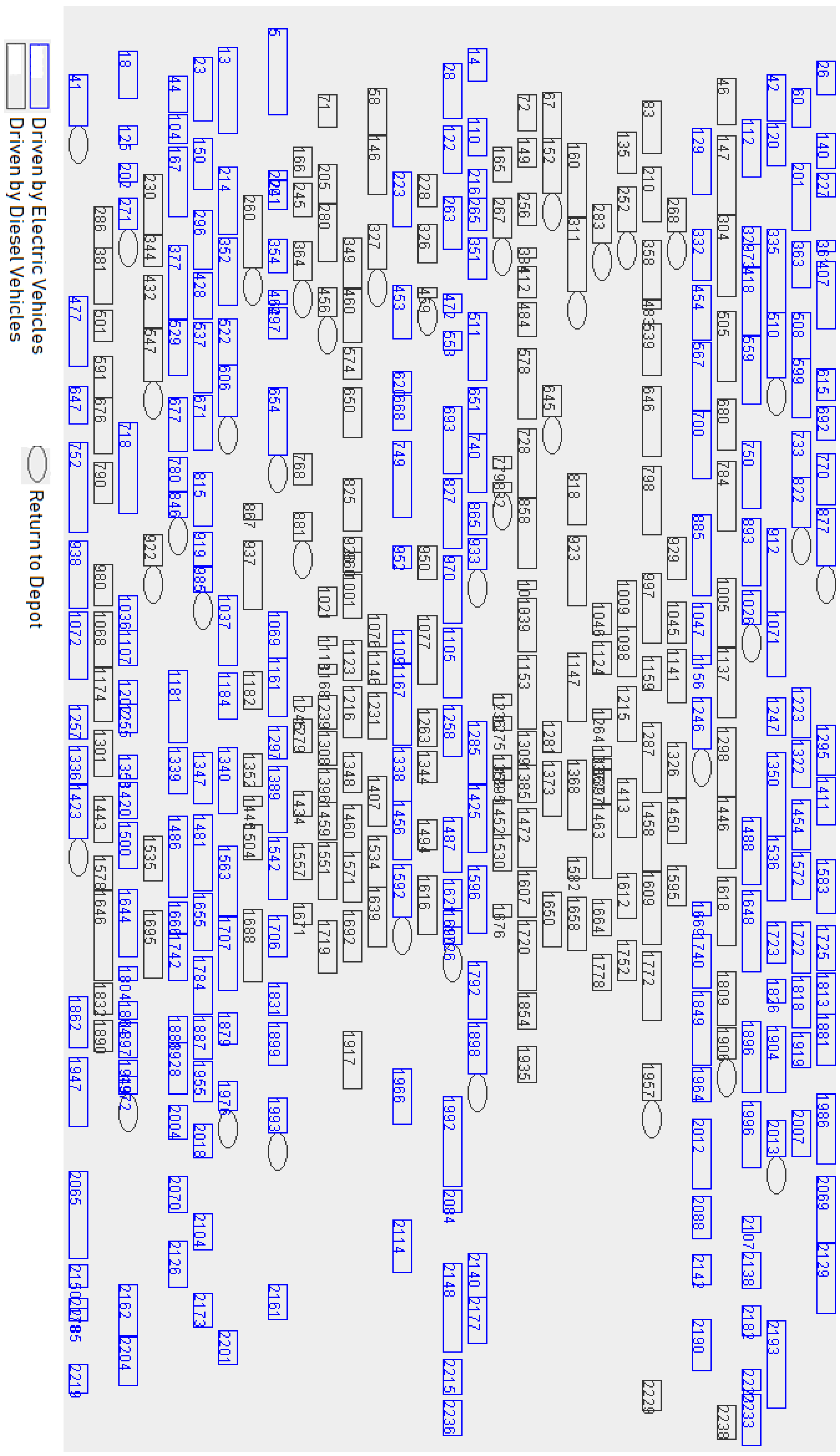


Figure 10: Part of the final schedule

To visualize the solutions resulting from the heuristic we have developed a tool which prints the results. In Figure 10 we show a part of the schedule based on historic data using 95% confidence values. Each row in this schedule represents a path which is driven by one single bus. We have made a distinction between diesel buses (black boxes) and electric vehicles (blue boxes). There are two observations to be made: there are only a few 'large' gaps between trips (besides recharging) for electric vehicles, which indicates that the vehicles are used efficiently. Secondly there are larger gaps between trips for diesel vehicles. This indicates that when a trip can be driven by a smaller (cheaper) bus, it will be driven by a smaller bus.

A large proportion of the total costs is due to renting additional vehicles, which are required to make a feasible solution. In Alkmaar we have to rent 13 large diesel buses, 1 standard diesel bus and 1 large electric bus. For the depot in Den Helder, 4 additional large diesel buses were necessary. Lastly, for the depot in Hoorn 13 large diesel buses had been rented.

Thus, the total number of additional buses rented is 32 - which costs 16.000 euros in this model. This number is due to the morning peak-hours for which all the large diesel buses are needed at the same time. This result is exaggerated when we are using the 95% confidence values for trip occupancy rates. However, we are using the schedules based on historic demand for each day of the week throughout the year, hence it is necessary to have a sufficient number of buses available.

In Table 6 the results are shown for different percentages of confidence. In this table, the objective values (Cost) and the total solving time in seconds are shown. Also the number of additional vehicles rented is viewed in this table. Additionally, we have stated the total time spent in the master problem and sub-problems (in seconds). Lastly, the approximation of the LP-lower bound (z^{LP}) is shown.

Naturally, we expect to see the number of additional vehicles needed drop for lower values of confidence. Though, we are interested in the trade-off. We have used the same initialization as described earlier.

Table 6: Results: Schedules using various confidence values

Confidence	Cost	Total Time	# Buses	Time in RMP	Time in SP	z^{LP}
97.5	33824	20490	38	20020	464	30503
95	30471	29997	32	29386	606	27507*
90	28019	24716	28	24094	619	25754
80	23109	35601	20	34887	710	22229

The most striking conclusion from these schedules is the amount of buses which are required to solve the problem. Even with 80% confidence values for trip occupancy. Again, this is due to the morning peak hours where almost all these large buses are needed simultaneously. The main factor to take into account is the capacity which can be deployed for trips during peak-hours.

Currently, the trip occupancy cannot exceed the number of seats + 50% of places to stand. What would happen if we were allowed to use the full capacity of the bus for trips in peak hours, instead of the number of seats + 50% of the places to stand?

In Table 7 the results are shown for different confidence values. Each problem has the same initialization as described earlier, using $Z_{MIN} = 0.01$ and $\theta_{MIN} = 0.70$.

Table 7: Full Capacity for trips in peak-hours

Confidence	Cost	Total Time	# Buses	Time in RMP	Time in SP	z^{LP}
97.5	20144	33912	15	32513	1393	18922
95	18904	41293	11	40408	881	16853
90	17904	46063	9	45175	884	15448
80	15726	64183	6	63066	1067	13544

In this case we see a sharp decrease in additional buses required, which is of course desirable. There are still some additional buses required to solve the problem. However, we think that the capacity restriction for trips in peak-hours should be revisited. It makes quite a difference whether you need 32 additional vehicles for the schedule with 95% confidence values or 11 additional vehicles.

7.4 Day-specific Schedules

In this section we will evaluate the performance of the heuristic using the two test-weeks as described in chapter 6. We want to emphasize on the choice of parameter Z_{MIN} and the trade-off in computation time and quality of the final solution.

For both weeks, we are solving the problem with the day-specific trip occupancy. Since the time available to solve the problem is an important aspect to keep in mind (when using forecasts, which are usually available one day earlier) we have less than a day to solve the problem. Therefore, the choice of Z_{MIN} is important! Thus, we will use 3 different values for Z_{MIN}. The results are shown in Tables 8 and 9. In these tables we show the total cost of the final schedule using our heuristic, the total computation times and the number of additional buses required to solve the problem.

Table 8: Results January 2017

	January								
Z_{MIN}	0.05			0.025			0.01		
Weekday	Cost	Time	#Buses	Cost	Time	#Buses	Cost	Time	#Buses
Monday	18367	20408	6	17517	33414	5	—	—	—
Tuesday	21308	14929	9	19249	22871	11	15632	53884	5
Wednesday	18269	16287	7	17334	29619	7	—	—	—
Thursday	18463	3231	5	16511	27073	5	14054	52922	7
Friday	17998	19168	8	14891	45961	3	—	—	—

Table 9: Results September 2017

	September								
Z_{MIN}	0.05			0.025			0.01		
Weekday	Cost	Time	#Buses	Cost	Time	#Buses	Cost	Time	#Buses
Monday	27968	5511	23	26637	12296	23	24511	33140	22
Tuesday	22287	10931	14	21376	19242	14	19876	42809	13
Wednesday	21258	12806	12	19919	25180	12	18738	49446	12
Thursday	22978	10373	15	21448	21304	15	19956	45211	14
Friday	21694	13874	13	20696	19168	13	19461	52652	13

Again, we see that quite a few additional large diesel buses are required for each weekday, especially for September. This result is a bit surprising, we would expect that the bus-fleet presented in Table 3 would be sufficient in the first place. But then again this result is due to the early peak hours. We cannot fully deploy the capacity of the vehicles. Therefore, all large vehicles are needed at the same time.

The costs for each day are significantly smaller when using day-specific trip occupancy rates than the results found for the schedules based on historical demand. This is mostly due to the costs of renting additional buses. The second explanation is that there are less large diesel vehicles required. Thus, less distance is covered by these vehicles which results in a decrease in operational costs.

New fleet

To make the decrease in operational costs more insightful we make the decision of 'buying' all additional buses required for the 95%-schedule which we have discussed in the last section and solve the problems again. These results are shown in Tables 10 and 11.

Table 10: Results New Fleet: January 2017

	January								
Z_{MIN}	0.05			0.025			0.01		
Weekday	Cost	Time	#Buses	Cost	Time	#Buses	Cost	Time	#Buses
Monday	15631	11703	0	14245	25681	0	12484	59639	0
Tuesday	15587	21532	0	13743	44819	0	12517	57353	0
Wednesday	15621	12031	0	13937	26739	0	12552	58102	0
Thursday	15581	20033	0	13914	45668	0	—	—	—
Friday	15466	10612	0	13890	26073	0	12424	62612	0

The heuristic - using Z_{MIN} as minimum relative decrease compared to the previous iteration - seems to be unstable for larger values of Z_{MIN}. The early termination criterion may be 'accidentally' reached with a relatively small decrease in objective value. This results in fixing the variables too fast and we will end up with a bad solution. This explains the outlier of Friday with $Z_{MIN} = 0.05$ in Table 11. This also explains the pattern of significantly lower running times in Table 10 for Monday, Wednesday and Friday.

Table 11: Results New Fleet: September 2017

	September								
Z_{MIN}	0.05			0.025			0.01		
Weekday	Cost	Time	#Buses	Cost	Time	#Buses	Cost	Time	#Buses
Monday	—	—	—	—	—	—	13178	54013	1
Tuesday	14667	19916	0	13304	43816	0	12512	56616	0
Wednesday	14896	20476	0	13422	45554	0	12234	56784	0
Thursday	14965	11773	0	13614	22286	0	12461	59824	0
Friday	18403	2419	4	13398	45236	0	12176	58703	0

Setting the parameter Z_{MIN} low enough seemingly prevents this problem. The computation time and end-results are similar for each weekday for both test-weeks using $Z_{MIN} = 0.01$. A different way to (partially) overcome this problem is to compare the relative decrease to I iterations ago. Lastly, we could use the absolute decrease similar to (Pepin et al. 2009), though as described earlier we think it is much less informative.

7.5 Discussion of the Results

We have shown that solving the problem on a daily basis - resulting in a day-specific schedule - leads to a significant reduction in operational costs (solely for operating buses). The heuristic provides stable results for parameter value $Z_{MIN} \leq 0.01$ in terms of quality and computation times.

We have consistently used parameter $\theta_{MIN} = 0.7$. Rarely we have seen multiple paths being fixed at the same time. And if it did, it has happened - without exception - in the first time we have fixed the results. Therefore, we would recommend looking only at lower values for θ_{MIN} for future research.

The computation times in Tables 10 and 11 are significantly higher for every single instance compared to the same problems in Tables 8 and 9. The increase in total running time is due to a significant drop in total costs. Since we are having a smaller objective value during the heuristic, the minimum required relative decrease is smaller than for larger objective values. This results in the early termination criterion not being met as soon as before. If we are using a stopping criterion similar to (Pepin et al. 2009) - using a minimum **absolute** decrease - the computation times would be more stable.

To conclude this chapter, we want to answer our research question: *"What is the benefit of making a vehicle schedule every day using the forecasts for trip occupancy?"* We are comparing the schedule based on historic data discussed in 7.3 with the day-specific schedules we have found in this section.

We want to emphasize on the fact that this is not really a fair comparison. The schedules based on historic demand can be deployed every day and have uncertain trip occupancy for each trip. This may lead to passengers having to stay behind at the bus-stops. For the day-specific schedules we have assumed that we know beforehand how many passengers will be travelling for each trip - and these schedules are made every day. However, this gives insight about the potential of the new method of Connexxion compared to the old method.

The schedule based on historic demand with 95% confidence values using $Z_{MIN} = 0.005$ resulted in a solution with objective value 29996. By subtracting the costs of renting additional buses, the resulting operational cost is $29996 - 32 \cdot 500 = 13996$. We compare the results using $Z_{MIN} = 0.01$ from Tables 10 and 11. We find for using $Z_{MIN} = 0.01$ or lower, the total costs of operating the same bus-fleet can be significantly reduced. On average we find a reduction of **9.3%** in operational costs compared to the schedule based on historic demand.

8 CONCLUSION & FUTURE RESEARCH

In this thesis we have formulated a column generation heuristic to solve large instances of the Multi-Depot Vehicles Scheduling Problem with electric vehicles.

The introduction of the electric fleet has let to one major challenge: Every electric vehicle has a limited action radius due to the battery. We want to make sure every vehicle makes a feasible path - without fully depleting the batteries and have a tow-truck to return the vehicle to the depot.

Additionally, for a path to be feasible we have had to make sure that every passenger can travel with the bus being deployed. Since the electric vehicles are smaller in terms of passenger capacity this problem has become more relevant.

To incorporate these additional problems in the vehicle schedule problem, we have proposed a capacity restriction on arcs in the network. Given the number of passengers present at the busiest moment during a specific trip, we can determine which types of buses can be deployed. We have also introduced the set of recharge arcs to facilitate recharging for electric vehicles. The heuristic uses this new network and provides stable results within a day for large instances of the MD-VSP using a partially electric fleet.

However, while the heuristic provides useful solutions, it leaves room for improvement. The algorithm used to solve the pricing problem for instance. As for now we have an algorithm which finds a path in less than 0.01 seconds, but it is not the optimal path for electric vehicles.

Furthermore, we have chosen not to include the additional costs which arise from using batteries as it is outside the scope of this thesis. The first issue is the lifespan of a battery. By completely draining a fully charged battery will result in a shorter lifespan of the battery, opposed to partially draining the battery and recharge it for a while. Therefore, the battery has to be replace it sooner. The second issue is that the costs of electricity can vary during the day. Therefore, recharging batteries may be cheaper in the night than during the day.

Lastly, we have seen large differences in the time spent solving the Restriced Master Problem and time spent solving the Pricing Problems. Even when the Restricted Master Problem is a relatively easy problem, a lot of time consumed solving it. This means that we are solving the Master Problem rather inefficiently, starting from scratch every single iteration. Providing a warm-start solution from the previous iteration to the new Restricted Master Problem may reduce the time needed to solve the RMP significantly.

We have a few interesting conclusions for the case study presented in this thesis. In the first place, we have shown that using the new bus-fleet for the old concession results in a lot of additional vehicles required to make sure every trip in the timetable can be driven. If we were able to fully use the passenger capacity of each bus during the morning peak hours, the number of additional buses is significantly reduced.

Secondly, we have shown that there is quite some potential for reducing the operational costs by making a new vehicle schedule every day rather than solving a schedule based on historical data. However, the next problem is how to realize this potential. The most important question would be *"how to deploy a new vehicle schedule every day"*?

By solving a vehicle schedule every day, we also have to assign personnel to duties to drive these buses around. Having a different route schedule every day means different duties for personnel every day. Planning new duties every day would make this problem much more complicated due to labour agreements and additional constraints in shifts. So, there is quite a lot to think about before we can use the new method.

There are several other uses for the heuristic we have developed. It can be used to calculate operational costs of timetables (and compare differences made in timetables) and calculate the amount of buses required to execute this timetable within reasonable time - and can be even faster than it is now.

To conclude this thesis: the way to proceed at this point of time is to develop a vehicle schedule using the forecasting method for specific scenarios. Having a schedule for a rainy day in summer or a day with snow in winter would incorporate the most important findings of any forecasting tool. The personnel planning problem is not as pressing as when we are solving a vehicle schedule every single day.

9 APPENDIX

9.1 A: Buslines

Lijn	Route	Type
1	Alkmaar Station - MCA - De Hoef - Alkmaar Station	
2	Alkmaar Overdie - Station - Daalmeer	
3	Alkmaar Station - Daalmeer	
4	Alkmaar Overdie - Station - Daalmeer	
5	Alkmaar Station - De Hoef	
6	Bergen - Alkmaar Station - Heerhugowaard	
9	Alkmaar Station - Oudkarspel	
10	Alkmaar Station -Oudorp-Langedijk-Heerhugowaard	
123	Alkmaar Station - De Rijp - Alkmaar Station	
129	Alkmaar Station - Noordbeemster - P'rend Tramplein	
150	Alkmaar Station - Nieuwe Niedorp - Schagen Station	
151	Alkmaar Station - Julianadorp	
155	Alkmaar Station - St. Maartenszee	
157	Alkmaar Station - Tuitjenhorn - Schagen Station	
160	Alkmaar Station - Heerhugowaard	
163	Alkmaar Station - Akersloot - Uitgeest	
164	Castricum Station - Egmond Binnen - Egmond aan Zee	
165	Alkmaar Station - Egmond a/Zee	
167	Castricum Station - Alkmaar Station	
N60	Alkmaar Station - Heerhugowaard	nacht
N69	Amsterdam Marnixstraat - Alkmaar Station	nacht
408	Heiloo 't Loo - Egmond aan Zee	buurtbus
409	Obdam - Ursem - Heerhugowaard	buurtbus
410	Egmond aan Zee - Schoorl	buurtbus
606	Alkmaar Station - Bergen Europese School	scholierenlijn
660	Alkmaar Station - Alkmaar Hogeschool	scholierenlijn
868	Castricum - Castricum aan Zee	strandlijn

Lijn	Route	Type
28	Texel Veerhaven - Den Burg - De Koog	
30	Den Helder Station - Station Zuid - Julianadorp	
31	Den Helder Station Zuid - Den Helder Station	
32	Den Helder Station - Station Zuid - Julianadorp	
33	Den Helder Station - Veerhaven	
34	Den Helder Station - De Schooten	
828	Texel Veerhaven - Den Burg - De Koog	
152	Schagen Station - Julianadorp	
158	Anna Pauwlowna - Den Helder Station	
250	Middenmeer - Alkmaar Station	
406	Schagen Station - Zijdewind - Waarland	buurtbus
411	Tuitjenhorn - Schagen Station	buurtbus
416	Kreileroord - Schagen Station	buurtbus
417	Schagen Station - Obdam	buurtbus
652	Den Helder Station - Schagen Station	scholierenlijn
653	Kreileroord - Wieringerwerf - Middenmeer - Schagen	scholierenlijn
658	Schagen Station - Anna Paulowna Station	scholierenlijn
851	Den Helder - Petten	strandlijn

Lijn	Route	Type
11	Hoorn Station - Risdam - Oostwoud	
12	Hoorn Station - Wognum - Opmeer - Hoogwoud	
13	Hoorn Station - Zwaag - Kersenboogerd	
14	Hoorn Station - Kersenboogerd - Hoorn 80	
128	Hoorn Station - Noordbeemster	
131	Hoorn Station - Blokker - Wervershoof - Andijk	
132	Hoorn Station - Wervershoof - Andijk	
134	Hoorn Station - Abbekerk Dorp - Wieringerwerf	
135	Hoorn Station - Wieringerwerf - Den Helder	
139	Hoorn Station - Abbekerk - Opperdoes - Medemblik	
239	Hoorn Station - Abbekerk - Opperdoes - Medemblik	
412	Hoorn Station - Hem - Hoogkarspel - Bovenkarspel	buurtbus
415	Medemblik - Wervershoof - Westwoud - Hoorn Station	buurtbus
438	Enkhuizen - Bovenkarspel - Wervershoof	buurtbus
617	Hoogwoud - Obdam	scholierenlijn
636	Aartswoud - Hoogwoud - Hoorn Station	scholierenlijn
650	Lelystad - Enkhuizen - Wervershoof	scholierenlijn

Nr	Route
701	Abbekerk Busstation - Abbekerk Gemeentehuis
702	Abbekerk Busstation - Benningbroek
703	Abbekerk Busstation - De Weere
704	Abbekerk Busstation - Lambertschaag
705	Abbekerk Busstation - Sijbekarspel
706	Abbekerk Busstation - Twisk West
707	Abbekerk Busstation - Twisk Oost
708	Anna Paulowna - Oude Sluis
709	Anna Paulowna - Wieringerwaard
710	Hoorn Station - Hensbroek
713	Hoorn Station - Spierdijk
714	Hoorn Station - Wogmeer
715	Hoorn Station - Zuidermeer
716	Schagen - Barsingerhorn-West
717	Schagen - Barsingerhorn-Oost
718	Schagen - Dirkshorn
719	Schagen Station - Crematorium
720	Schagen - 't Veld
721	Schagen - Kolhorn
722	Schagen - Lutjewinkel
723	Schagen - Sint Maarten
724	Schagen - Sint Maartensbrug
725	Schagen - Stroet
726	Schagen - Zijdewind
727	Wieringerwerf - Kreileroord
728	Hoorn Station - Midwoud
729	Hoorn Station - Oostwoud
730	Hoorn Station - Nibbixwoud
731	Hoorn Station - Oosterblokker
732	Hoorn Station - Westwoud
733	Hoorn Station - Zwaagdijk
734	Alkmaar Station - St. Pancras

9.2 B: Recharge Function

Let us define the following parameters:

- $SoC[v]$ the state of charge at node v
- $d_{v,depot}$ the distance travelled from node v to the depot
- γ_{vu} the total recharge time available using arc (v,u)
- R the action radius

Algorithm 3: Calculating SoC^+

```
Data: $SoC[v], d_{v,depot}, \gamma_{vu}, R$
Result: $SoC^+$
if $SoC[v] - d_{v,depot} < R \cdot 0.8$ then
    $t_{80} = \frac{R \cdot 0.8 - SoC[v] - d_{v,depot}}{R \cdot 0.8} \cdot 60;$
    $t_{100} = t_{80} + 60;$
    if $\gamma_{vu} < t_{80}$ then
        $SoC^+ = \frac{R \cdot 0.8 - SoC[v] - d_{v,depot}}{R \cdot 0.8} \cdot \gamma_{vu}$
    end
    if $t_{80} \leq \gamma_{vu} \leq t_{100}$ then
        $SoC^+ = \frac{R \cdot 0.8 - SoC[v] - d_{v,depot}}{R \cdot 0.8} \cdot t_{80} + \frac{R \cdot 0.2}{60} \cdot (\gamma_{vu} - t_{80})$
    end
    if $\gamma_{vu} > t_{100}$ then
        $SoC^+ = R$
    end
else
    $t_{100} = \frac{R - SoC[v] - d_{v,depot}}{R \cdot 0.2} \cdot 60;$
    if $\gamma_{vu} < t_{100}$ then
        $SoC^+ = \frac{R \cdot 0.2}{60} \cdot \gamma_{vu}$
    end
    if $\gamma_{vu} \geq t_{100}$ then
        $SoC^+ = R$
    end
end
```

REFERENCES

Adler, J.D. and P.B. Mirchandania (2017). "The Vehicle Scheduling Problem for Fleets with Alternative-Fuel Vehicles". In: *Transportation Science* 51.2, pp. 441–456.

Bertossi, A.A., P. Carraresi, and G. Gallo (1987). "On some Matching Problems Arising in Vehicle Scheduling Models". In: *Networks* 17.3, pp. 171–281.

Bruglieri, M. et al. (2015). "A Variable Neighborhood Search Branching for the Electric Vehicle Routing Problem with Time Windows". In: *Electronic Notes in Discrete Mathematics* 47, pp. 221–228.

Carrareri, P. and G. Gallo (1984). "Network models for vehicle and crew scheduling". In: *European Journal of Operational Research* 16.2, pp. 139–151.

Chao, Z. and C. Xiaohong (2013). "Optimizing battery electric bus transit vehicle scheduling with battery exchanging: model and case study". In: *Procedia - Social and Behavioral Sciences* 96.6, pp. 2725–2736.

Daduna, J.R. and J.M.P. Paixão (1995). "Vehicle Scheduling for Public Mass Transit". In: *Computer-Aided Transit Scheduling* 430, pp. 76–90.

Desrosiers, J. and M.E. Lübbecke (2005). "A Primer in Column Generation". In: *Desrosiers, J., Solomon, M.M., Desaulniers, G. (Eds.), Column Generation*. Springer.

Ferland, J. and P. Michelon (1988). "The Vehicle Scheduling Problem with Multiple Vehicle Types". In: *Journal of the Operational Research Society* 39.6, pp. 577–583.

Gintner, V., N. Kliewer, and L. Suhl (2005). "Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice". In: *OR Spectrum* 27, pp. 507–523.

Haghani, A. and M. Banihashemi (2002). "Heuristic approaches for solving large-scale bustransit vehicle scheduling problem with route time constraints". In: *Transportation Research Part A: Policy and Practice* 36.4, pp. 309–333.

Huisman, D., R. Freling, and A. Wagelmans (2004). "A Robust Solution Approach to the Dynamic Vehicle Scheduling Problem". In: *Transportation Science* 38.4, pp. 447–458.

Kliewer, N. and S. Bunte (2009). "An overview on vehicle scheduling models". In: *Journal of Public Transport* 1.4, pp. 299–317.

Kliewer, N., T. Mellouli, and L. Suhl (2006). "A Time-Space Network Approach for the Integrated Vehicle- and Crew-Scheduling Problem with Multiple Depots". In: *Transportation Science* 44.3, pp. 367–382.

Kooten-Niekerk, M.E. van, J.M. van den Akker, and J. A. Hoogeveen (2017). "Scheduling electric vehicles". In: *Public Transportation* 9, pp. 155–176.

Naumann, S., L. Suhl, and S. Kramkowski (2011). "A stochastic programming approach for robust vehicle scheduling in public bus transport". In: *Procedia Social and Behavioral Sciences* 20, pp. 826–835.

Oukil, A., H.B. Amor, and G. Desaulniers (2006). "Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems". In: *Computers and Operations Research* 34, pp. 817–834.

Pepin, A.S. et al. (2009). "A comparison of five heuristics for the multiple depot vehicle scheduling problem". In: *Journal of Scheduling* 12, pp. 17–30.

Ribeiro, C. and F. Soumis (1994). "A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem". In: *Operations Research* 42, pp. 41–52.

Schneider, M., A. Stenger, and D. Goeke (2014). "The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations". In: *Transportation Science* 48.4, pp. 500–520.

Wang, H. and J. Shen (2007). "Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints". In: *Applied Mathematics and Computation* 190, pp. 1237–1249.