Optimizing for Reliable and Sustainable Public Transport

Marcel van Kooten Niekerk

2018

Marcel van Kooten Niekerk

Optimizing for Reliable and Sustainable Public Transport

ISBN: 978-90-393-6970-8

Cover Illustration: Twan Beurskens - Studio Luidspreker - Venlo (NL)
Cover Design: Print Service Ede - Ede (NL)
Printed by: Print Service Ede - Ede (NL)

Optimizing for Reliable and Sustainable Public Transport

Optimaliseren voor Betrouwbaar en Duurzaam Openbaar Vervoer
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus prof.dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 30 mei 2018 des middags te 2.30 uur

door

Marcel Erik van Kooten Niekerk

geboren op 17 juni 1974 te Haarlem

Promotor:	Prof.dr. J. van Leeuwen
Copromotoren:	Dr.ir. J.M. van den Akker
	Dr. J.A. Hoogeveen

This research was generously supported by Qbuzz BV.

Preface

Before you lies the result of my pursuit to make public transport better. This pursuit has a long history and will not end with this thesis.

In 2010 I finished my master's thesis on cyclic crew scheduling (see van Kooten Niekerk [2010]), the first subject in public transport which I studied at length. At that time, I already started working for Qbuzz, a new public transport company founded in 2008. My job was to optimize the exploitation of the public transport and to contribute to public tenders of public transport in areas in the Netherlands. For this I used commercially available software, but soon I noticed that this software did not always meet our requirements, so I saw opportunities for further optimizations.

I was intrigued by the complexity of the optimization problems in this research field. So after graduating I looked for an opportunity to improve my knowledge of optimization. For this, an excellent scientific basis was necessary. The obvious answer was that I needed to continue my contacts with scientific research, which could be achieved by becoming a PhD-student at Utrecht University. So I proposed this to Leon Struijk and Rob van Holten, the founders of Qbuzz, and I was happy they agreed with me and that they were willing to support this. I am grateful to Utrecht University, especially to the Department of Information and Computing Sciences, for accommodating my PhD-project.

The subjects that are studied in this thesis all originate from core issues in public transport. Some subjects arose from public tenders, where there was usually only a few weeks time to develop the first working prototype, others originate from my own ideas to improve public transport in general. During my PhD-research I could freely explore my ideas for new methods and solutions and experiment with them to determine the approaches that would work best in practice. All subjects in this thesis have in common that the research is already in use and applied at Qbuzz.

The completion of this PhD-thesis will not end my pursuit of a more optimal public transport. I am convinced that new optimization methods will be a relevant and necessary ingredient of this pursuit. Also, faster computers, the increasing amount of available data and newly developed algorithms will make continuous research necessary.

Completing a PhD-thesis is not a process that can be completed without discussion with and help from others. First of all I want to thank my daily supervisors. First Marjan van den Akker, who I first met at the CASPT-conference in Leeds in 2006, where she told me about the work on public transport that is done at Utrecht University, This raised my interest in studying computer science and, in combination with motivations from my wife Tonny, it forms the base of this PhD-thesis. I thank Marjan for the many discussion we had on

modeling public transport and for the many things I learned from her about analyzing the models. I also want to thank my other daily supervisor Han Hoogeveen for his expert advice on many issues dealing with the use of optimization methods and their assessment, and for the delicious coffee he made, which was always better than the coffee from the coffee machine at the department. Last but not least I want to thank my promotor Jan van Leeuwen for the inspiring discussions we had, which led to new ideas and improvement of existing ideas, and for his guidance in the final stages of completing my thesis.

Furthermore, I want to thank Leon Struijk and Rob van Holten from Qbuzz for making it all possible and providing the conditions to do my PhD-research.

During the years of preparing my PhD-thesis, I discussed my research with a lot of people. I want to mention a few of them, that made an important contribution to the research. First of all I want to thank Leo Kroon and Gábor Maróti for the discussion we had about the determination of planned trip runtimes. Although we discussed it only once during two hours, it led to new insights and a major improvement in my research. I also want to thank Charles Fleurent from Giro for the numerous discussions on the algorithms that Giro uses in Hastus and those that I have developed. Furthermore, I want to thank Joost Swinnen from De Lijn, for providing timetable data of Leuven, which I used for my research on scheduling electric vehicles.

For the completion of the research I owe a lot to all my colleagues at Qbuzz, who provided data, gave feedback on the results, supported me and made Qbuzz a pleasant place to work. In particular, I want to thank Jan Kouwenhoven, Martin Kruis, Gerrit Spijksma, Job Teunissen, Daniël Vervoort and Dennis Vlugt.

And last but certainly not least, I want to thank my wife Tonny, without whom I would never have started studying again, for her relentless support and coaching during all the years and for all sacrifices she made in order to make my PhD a success.

Contents

Chapter 1

Introduction

Everywhere people want to move from A to B. Moving can be done by walking, taking the bike or car or using a taxi. If these measures are not available or the distance is too large to walk or cycle, public transport by train, tram or bus may supply the method to move. The goal of public transport is to bring people from their origin to their destination in an effective way. In order to be effective, public transport needs to have the following properties:

- *Fast* – Passengers should reach their destination as soon as possible. From the perspective of the passenger, the public transport ideally brings them straight to their destination, without detours and transfers.

- *Easy* – Knowing how to get from A to B should be easy, as well as the travel itself and paying for the trip.

- *Reliable* – Passengers should know in advance when they have to leave from A and when they arrive at B. Also, the vehicles should adhere to the planned schedule as much as possible so that passengers do not miss the vehicle or have to wait longer than planned. During periods when vehicles arrive and depart frequently, passengers may want to rely on the low average waiting times rather than on perfect compliance to the planned schedule.

- *Efficient* – A public transport system should be efficient, and cost effective. Resources like buses and drivers cost money and should be used efficiently while offering a high quality public transport network with respect to the earlier mentioned properties. A high quality public transport will also lead to higher revenues: As the attractiveness of public transport increases, the number of passengers will increase.

Beside these four properties, we add one more property that is emerging in the last decades:

- *Sustainable* – A public transport system should be environment-friendly. Where possible, the pollution caused by public transport should be reduced and green energy sources should be employed.

To make a public transport network efficient, a lot of planning has to be done. This planning leads to complex optimization problems. This thesis is devoted to key issues in the optimization of public transport with the focus on reliability and sustainability. For the optimization for reliability we use historical measurements of run times to develop models for evaluation and optimization of reliability of a public transport network. For the sustainability we include environmental concerns in the optimization: electric vehicles with no exhaust from the engine as well as a mixed fleet with electric vehicles and other vehicles of varying degrees of exhaust. We aim to develop models and optimization methods for these problems which can be used in practice. These methods should be able to handle real-life sized instances and should perform better than currently used algorithms or heuristics.

In the rest of this chapter we will describe the planning processes behind the creation of an efficient public transport network. In Section 1.1 we describe the vocabulary that comes with a public transport network, after which we describe the planning in more detail in Section 1.2. Planning an efficient public transport network gives many opportunities to optimize, in fact, most steps of planning public transport are optimizations. The various optimizations are described in Section 1.3, followed by a literature overview on these optimizations in Section 1.4. We end this introduction with a short description of our contribution to the optimization problems of public transport in Section 1.5.

1.1 Introduction to public transport

Designing a public transport network starts with determining the *routes* which the vehicles drive. We aim to bring the passengers to their destination as fast as possible, and if it is not possible to bring all passengers directly to their destination, then the goal is also to minimize the number of transfers needed.

A route is specified by a sequence of stops, i.e. locations where the vehicle will stop to embark and/or disembark passengers. On a route, a set of *trips* is planned, where a trip is one single travel of a vehicle along the route or a part of the route. For every trip, the planned departure and arrival time and start and end stop are defined. These trips will be published in a *timetable* for a route. See Figure 1.1 for an example of a timetable.

Lijn 68a van Winsum naar Leens

zon- en feestdagen

Ritnummer:		7002	7004	7006	7008	7010
Winsum, Station	V	8:17	9:17	10:17	11:17	12:17
Baflo, Heerestraat		8:26	9:26	10:26	11:26	12:26
Den Andel, Andelsterweg		8:29	9:29	10:29	11:29	12:29
Den Andel, Kerk		8:30	9:30	10:30	11:30	12:30
Westernieland, J Heidemastraat		8:34	9:34	10:34	11:34	12:34
Pieterburen, Pieterplein		8:38	9:38	10:38	11:38	12:38
Pieterburen, Wiebenerweg		8:40		10:40		12:40
Broek, Driesprong		8:45		10:45		12:45
Kleine Huisjes, Dijksterweg		8:47		10:47		12:47
Kloosterburen, Hoofdstraat 44		8:50		10:50		12:50
Hornhuizen, Kerk		8:56		10:56		12:56
Leens, De Nije Nering	A	9:02		11:02		13:02

Figure 1.1: Example of a timetable

For every stop, departure times of vehicles are planned and published in the timetable. The passenger can use this to find out at what time he should be at the stop. In order to be reliable, the vehicle should not pass the stop before the planned departure time, but in reality this is not always feasible. For example, when the stop is on a busy one-lane road, waiting at a stop when a vehicle is too early will block the traffic. Stops at which it is possible to wait are called *holding points*; the vehicles are held until it is time to leave according to the timetable.

The time needed to drive the trip is called the *trip runtime*. In this thesis, we will distinguish between planned runtime and measured runtime. Planned runtime is the runtime according to the timetable, measured runtime is the runtime measured in reality.

Since driving times in peak hours are usually different from those during e.g. the evening, the planned trip runtimes on a route may not have the same value during the whole day. The set of trips that do have an equal planned runtime during the day is called a *runtime group*.

In the timetable, we usually see that every n minutes a trip on a route will depart. This is called a *headway* of n minutes. When there are m trips per hour, we also speak about a *frequency* of m trips per hour.

In order to perform all trips in a timetable, the Public Transport Organization (PTO) will schedule vehicles on all trips. A vehicle typically will drive multiple trips a day. The list of trips that a vehicle is scheduled to do on one day is called a *vehicle task*. A vehicle task does not only contain *in-service trips*, i.e. trips from the timetable, but also trips that are necessary to get from and to the vehicle depot and to connect to trips that do not start at the place where the previous trip ended. These trips are called *deadhead trips*.

Most vehicles need a driver, so the drivers have to be planned as well. Because the constraints for drivers, e.g. because of labour regulations, are different from those for vehicles, we can not copy the planning for vehicles to the drivers. The work that a driver has to do during one day is called a *duty* and will consist of parts of the vehicle tasks which we call *duty pieces*. On top of this, a duty may also contain other activities like meal breaks and vehicle preparation.

1.2 Planning of public transport

When designing a public transport network, we first look at the need for transport. How many people are expected to use the transport network? What are the origins and destinations of their travels? From these data, we design a set of routes along which vehicles will drive. Given the expected number of passengers, we decide at which frequency a route will be driven and which vehicle type will be used.

Using these routes and frequencies, a timetable is developed, where for each trip all trip passing times along stops are determined. The goal of a timetable is to plan all trips, so that the passenger knows at what time the bus is planned to leave and arrive. In order to drive a timetable, buses and drivers are needed. They both cost money, so while developing a timetable, the PTO will also take these costs into account. The planning of a public transport network to obtain an ultimate schedule for vehicles and crew can be split in six phases:

- *Transit Network Design*. The first step is designing a transit network. Using the data about demand, the routes and desired connections between routes are determined. The

goal is to minimize the travel times of the passengers and the number of passengers that have to change during their journey and to keep the network simple. It should not be too difficult for passengers to figure out how to travel. Although we do not have exact trip durations yet at this stage, we can minimize the travel duration by reducing detours in routes when possible and by determining which connections are used the most by the passengers.

- *Trip runtime determination*. When we know the routes of the trips, we have to determine the timetable of such a route. Ideally, a vehicle is always on time, but because of the stochastic nature of traffic and passengers, this will not be the case. Using the measured times when a bus stops at a bus stop, we determine which times should be published as planned departure times at the stops in order to prevent early passages and to reduce the amount of delay by as much as possible, while the trip duration should also not become too long.

- *Timetable pattern*. When a route needs to be driven frequently, it may be desirable to plan the timetable so that every hour or twice an hour a bus will pass. When such a regular timetable is desired, the first step is to create a timetable for one hour, which is the template for the full timetable. We call such a template timetable a *timetable pattern*. In Table 1.1, two example timetable patterns are shown. The left one is for the daytime period (6:00 - 18:00[1]), the right one is for the evening period (after 18:00).

Daytime pattern Direction 1		
Stop	**Pat.1**	**Pat.2**
A	x:03	x:33
B	x:23	x:53
C	x:43	x:13

Evening pattern Direction 1	
Stop	**Pat.1**
A	x:03
B	x:23
C	x:43

Daytime pattern Direction 2		
Stop	**Pat.1**	**Pat.2**
C	x:47	x:17
B	x:07	x:37
A	x:27	x:57

Evening pattern Direction 2	
Stop	**Pat.1**
C	x:47
B	x:07
A	x:27

Table 1.1: Example timetable pattern. The first column of each table contains the stop names, the other columns the trip passing times.

[1] In this thesis we use an extended 24-hour notation, where we continue counting after midnight. Thus, 18:00 equals 6 pm, and 25:00 equals 1 am the next day.

- *Timetable*. In many cases, a timetable is built based on one or more timetable patterns. This is done by repeating the appropriate timetable pattern for every hour of a day. Different timetable patterns may be used for different parts of the day, when driving time or frequency changes. In Table 1.2, an example timetable is shown, based on the timetable patterns from Table 1.1. The first trip for both directions in the timetable is the first one that starts after 6:00 and the last trip is the first one that starts after midnight. In Figure 1.2, a time-distance graph of this timetable is drawn, with the time along the x-axis and the position along the y-axis.

- *Vehicle schedule*. In order to drive all trips from the timetable, we make a planning for the vehicles. Every trip should be in the plan of exactly one vehicle. One of the goals is efficiency; therefore, we should use as few vehicles as possible and minimize the total distance driven by these vehicles. Another goal is punctuality, which can be influenced here by including buffer times between trips, which are used to absorb the delay of the preceding trip. We should also take into account that the bus starts and ends its day in a depot, where it is parked overnight. For this, the bus has to drive empty to a starting point; this trip from or to the depot is called a *pull trip*. Sometimes an empty trip between two trips is necessary. Recall that these trips without passengers are called *deadhead trips*. For our example timetable, we need three buses. The work scheduled for one vehicle is called a *vehicle task*. In Figure 1.3, we have drawn the vehicle schedule from Table 1.3. The trips are represented by horizontal lines, with small vertical lines hanging below it that indicate their start- and end time. The dotted lines represent the pull trips and deadhead trips.

- *Crew schedule*. After creating a vehicle schedule, we need to plan drivers for these vehicles. The schedule should be divided in duties, which are pieces of work for one driver on one day that comply with all rules and legislation applicable. In our example, a driver may not work for more than 9 hours on a day and should get at least two short breaks or one long break during a duty. These duties are usually different from the vehicle tasks, because these constraints are not necessary for vehicles: they can drive all day long without break, as long as there is enough fuel. In Table 1.4, a set of 7 duties is shown covering the vehicle tasks from the example. In Figure 1.4, these duties are displayed graphically.

1.3 Optimization of public transport scheduling

A PTO wants to offer the best transportation which is fast, easy and reliable, but also cost-effective and efficient. These goals are usually contradictory. Therefore the objective in optimizing public transport is normally split in four different criteria.

- *Timetable quality*. A large source of revenues are the fares paid by the passengers. These revenues can be increased by changing the height of the fares or by attracting more people to the public transport by offering a good timetable. To optimize the

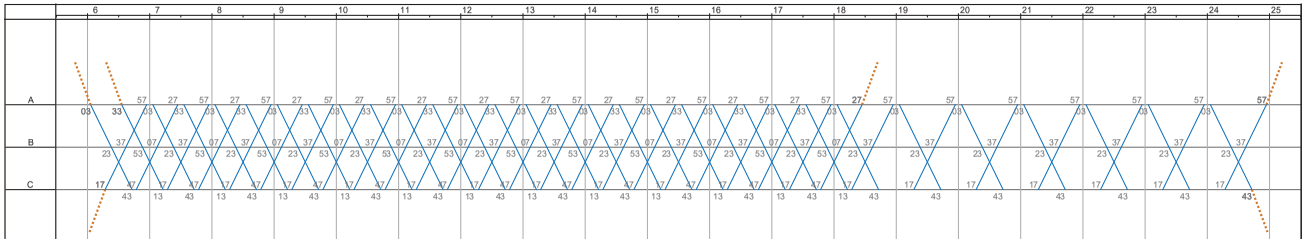


Figure 1.2: Graph representing the timetable from Table 1.2. On the x-axis, you see the time of day and on the y-axis you see one line for each place along the route. The numbers near the lines are the passing times at that stop.

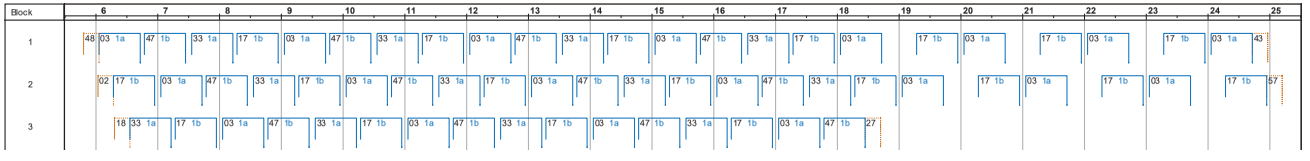


Figure 1.3: Graph representing the vehicle tasks from Table 1.3. On the x-axis, you see the time of day. Every line corresponds with one vehicle task. For example: when we look at the start of the first line, the first trip starts at 5:48. It is dotted, so it is a deadhead trip. The second trip starts at 6:03 and the text '1a' shows that it is line 1 in direction a. This trip corresponds to the line in Figure 1.2 that starts at A at 6:03 and ends at C at 6:43.

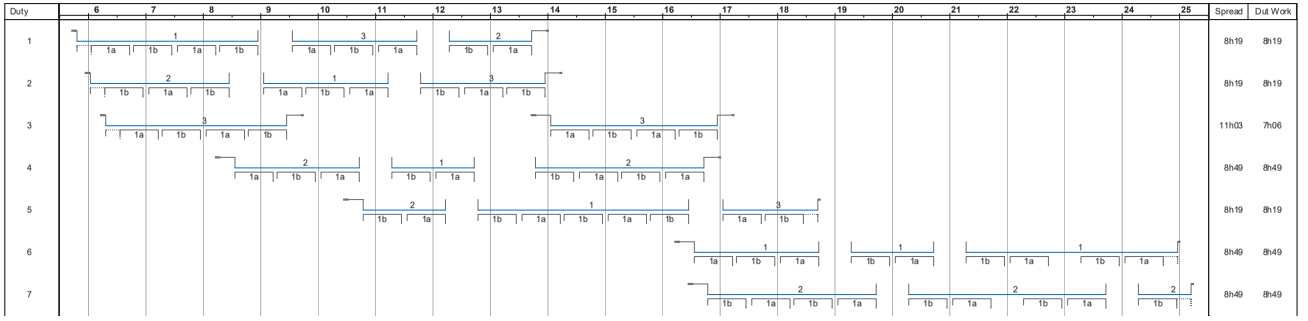


Figure 1.4: Graph representing the crew schedule from Table 1.4. On the x-axis, you see the time of day. Every line corresponds to a single crew duty. A duty consists of parts of the vehicle tasks in Figure 1.3. The upper part indicates the piece (or part of block) we use, the lower part shows the trips that this part of the block contains. On top of each piece we see the number of the vehicle task. The horizontal lines at the start or end of a piece indicate travels of the crew, for which obviously no piece is necessary.

Timetable Direction 1											
Stop	**Trip**	**Trip**	**Trip**	**Trip**		**Trip**	**Trip**	**Trip**	**Trip**		**Trip**
A	6:03	6:33	7:03	7:33	every	17:33	18:03	19:03	20:03	every	24:03
B	6:23	6:53	7:23	7:53	30 min	17:53	18:23	19:23	20:23	hour	24:23
C	6:43	7:13	7:43	8:13	until	18:13	18:43	19:43	20:43	until	24:43

Timetable Direction 2											
Stop	**Trip**	**Trip**	**Trip**	**Trip**		**Trip**	**Trip**	**Trip**	**Trip**		**Trip**
C	6:17	6:47	7:17	7:47	every	17:47	18:17	19:17	20:17	every	24:17
B	6:37	7:07	7:37	8:07	30 min	18:07	18:37	19:37	20:37	hour	24:37
A	6:57	7:27	7:57	8:27	until	18:27	18:57	19:57	20:57	until	24:57

Table 1.2: Example timetable

quality of a timetable, we need an objective function. Some possible components are average travel time per passenger, the frequency of a bus line, and the average waiting time for connections between two trips.

- *Punctuality*. The quality of service is also important for passengers. The time spent waiting for a bus to arrive should be minimized, as well as delays and the number of missed connections.

- *Vehicle cost*. We split the vehicle cost into two parts: the fixed cost and the variable cost. The fixed cost is the cost of a vehicle, which is independent of the distance driven by the vehicle, for example the depreciation of a vehicle per year and cost of insurance. The variable cost is the cost per driven kilometer. These include fuel cost and maintenance cost.

- *Crew cost*. The crew that is scheduled on the vehicles is a major cost factor for a PTO. Crew cost can also be split in fixed and variable costs. Examples of fixed costs are health insurance plans and pension plans. Variable costs are the cost per hour per driver for salary cost.

We now have defined six stages of optimization in Section 1.1 and four types of objectives in Section 1.3. In Table 1.5, every combination that has been investigated so far is marked with a representative reference.

This thesis is devoted to a series of problems that deal with the issues that we have identified and aims to improve over current methods. For the trip runtime determination and for vehicle scheduling our goal is to improve the reliability by using the large amounts of historical data that is currently available, together with algorithms that use this data. In Chapters 5 and 6, we aim to improve vehicle scheduling in such a way that the impact of the schedule on the environment is taken into account and reduced as much as possible.

Vehicle 1			
From	**To**	**Start**	**End**
Depot	A	5:58	6:03
A	C	6:03	6:43
C	A	6:47	7:27
A	C	7:33	8:13
C	A	8:17	8:57
A	C	9:03	9:43
C	A	9:47	10:27
A	C	10:33	11:13
C	A	11:17	11:57
A	C	12:03	12:43
C	A	12:47	13:27
A	C	13:33	14:13
C	A	14:17	14:57
A	C	15:03	15:43
C	A	15:47	16:27
A	C	16:33	17:13
C	A	17:17	17:57
A	C	18:03	18:43
C	A	19:17	19:57
A	C	20:03	20:43
C	A	21:17	21:57
A	C	22:03	22:43
C	A	23:17	23:57
A	C	24:03	24:43
C	Depot	24:43	25:13

Vehicle 2			
From	**To**	**Start**	**End**
Depot	C	5:47	6:17
C	A	6:17	6:57
A	C	7:03	7:43
C	A	7:47	8:27
A	C	8:33	9:13
C	A	9:17	9:57
A	C	10:03	10:43
C	A	10:47	11:27
A	C	11:33	12:13
C	A	12:17	12:57
A	C	13:03	13:43
C	A	13:47	14:27
A	C	14:33	15:13
C	A	15:17	15:57
A	C	16:03	16:43
C	A	16:47	17:27
A	C	17:33	18:13
C	A	18:17	18:57
A	C	19:03	19:43
C	A	20:17	20:57
A	C	21:03	21:43
C	A	22:17	22:57
A	C	23:03	23:43
C	A	24:17	24:57
A	Depot	24:57	25:02

Vehicle 3			
From	**To**	**Start**	**End**
Depot	A	6:28	6:33
A	C	6:33	7:13
C	A	7:17	7:57
A	C	8:03	8:43
C	A	8:47	9:27
A	C	9:33	10:13
C	A	10:17	10:57
A	C	11:03	11:43
C	A	11:47	12:27
A	C	12:33	13:13
C	A	13:17	13:57
A	C	14:03	14:43
C	A	14:47	15:27
A	C	15:33	16:13
C	A	16:17	16:57
A	C	17:03	17:43
C	A	17:47	18:27
A	Depot	18:27	18:32

Table 1.3: Example vehicle schedule. Every line corresponds with one trip. For every trip, only the place and time at the start and end of the trip are given.

Duty 1. Regular Duty				
Activity	**From**		**To**	
Sign on	5:42	Depot	5:48	Depot
Veh 1	5:48	Depot	8:57	A
Break	8:57	A	9:33	A
Veh 3	9:33	A	11:43	C
Break	11:43	C	12:17	C
Veh 2	12:17	C	13:43	C
Travel	13:43	C	13:58	Depot
Sign off	13:58	Depot	14:01	Depot

Duty 2. Regular Duty				
Activity	**From**		**To**	
Sign on	5:56	Depot	6:02	Depot
Veh 2	6:02	Depot	8:27	A
Break	8:27	A	9:03	A
Veh 1	9:03	A	11:13	C
Break	11:13	C	11:47	C
Veh 3	11:47	C	13:57	A
Travel	13:57	A	14:12	Depot
Sign off	14:12	Depot	14:15	Depot

Duty 3. Split Duty				
Activity	**From**		**To**	
Sign on	6:12	Depot	6:18	Depot
Veh 3	6:18	Depot	9:27	A
Travel	9:27	A	9:42	Depot
Sign off	9:42	Depot	9:45	Depot
Sign On	13:42	Depot	13:48	Depot
Travel	13:48	Depot	14:03	A
Veh 3	14:03	A	16:57	A
Travel	16:57	A	17:12	Depot
Sign off	17:12	Depot	17:15	Depot

Duty 4. Regular Duty				
Activity	**From**		**To**	
Sign on	8:12	Depot	8:18	Depot
Travel	8:18	Depot	8:33	A
Veh 2	8:33	A	10:43	C
Break	10:43	C	11:17	C
Veh 1	11:17	C	12:43	C
Break	12:43	C	13:47	C
Veh 2	13:47	C	16:43	C
Travel	16:43	C	16:58	Depot
Sign off	16:58	Depot	17:01	Depot

Duty 5. Regular Duty				
Activity	**From**		**To**	
Sign on	10:26	Depot	10:32	Depot
Travel	10:32	Depot	10:47	C
Veh 2	10:47	C	12:13	C
Break	12:13	C	12:47	C
Veh 1	12:47	C	16:27	A
Break	16:27	A	17:03	A
Veh 3	17:03	A	18:42	Depot
Sign off	18:42	Depot	18:45	Depot

Duty 6. Regular Duty				
Activity	**From**		**To**	
Sign on	16:12	Depot	16:18	Depot
Travel	16:18	Depot	16:33	A
Veh 1	16:33	A	18:43	C
Break	18:43	C	19:17	C
Veh 1	19:17	C	20:43	C
Break	20:43	C	21:17	C
Veh 1	21:17	C	24:58	Depot
Sign off	24:58	Depot	25:01	Depot

Duty 7. Regular Duty				
Activity	**From**		**To**	
Sign on	16:26	Depot	16:32	Depot
Travel	16:32	Depot	16:47	C
Veh 2	16:47	C	19:43	C
Break	19:43	C	20:17	C
Veh 2	20:17	C	23:43	C
Break	23:43	C	24:17	C
Veh 2	24:17	C	25:12	Depot
Sign off	25:12	Depot	25:15	Depot

Table 1.4: Example Crew Schedule

We optimize vehicle schedules for electric vehicles as well as for mixtures of vehicles with different exhaust properties.

We have also added our research from Chapters 3 and 4 to the table. Chapters 5 and 6 are not in this table, as we view these as extensions to vehicle scheduling.

	Timetable quality	**Punctuality**	**Vehicle cost**	**Crew cost**
Transit network design	van Nes [2002]		van Nes [2002]	
Trip runtime determination	Chapter 3	van Oort [2011], Kroon et al. [2007], Chapter 3		
Timetable pattern	Peeters [2003]		Peeters [2003]	van Kooten Niekerk [2010]
Timetable	Bruno et al. [2009]			
Vehicle schedule		Amberg et al. [2011], Chapter 4	Huisman [2004]	
Crew schedule		Amberg et al. [2011]		

Table 1.5: Stages of optimization vs. objectives

1.4 Literature overview

In this section, we will briefly discuss the literature mentioned in Table 1.5. References to chapters in this thesis are described in Section 1.5.

The PhD-thesis of van Nes [2002] deals with the design of multimodal transportation networks, taking all kinds of transport into account. Chapter 6 of his PhD-thesis deals with public transport networks. In this chapter, the author makes a mathematical model of a public transport network. Using this model, he optimizes a public transport network for operational cost and passenger welfare by changing spacing of stops, changing frequency and changing modality of the transport. He also evaluates multi-modal public transport networks.

In van Oort [2011], the author shows a way to optimize the trip runtimes during stops, and investigates the influence it has on punctuality. He classifies the stops in two categories: important stops and non-important stops. For both categories he takes a fixed percentile-value. With these values, he calculates the average delay of the trips. Then he optimizes the average delay by varying these percentile-values.

Kroon et al. [2007] also determines optimal trip runtimes. They do this for a cyclic timetable of a train network. Assuming that trains never leave too early at a station, they determine the optimal trip departure times at the stations while taking into account all rail-specific constraints like track occupations and switches. This is done by modeling the train network as a MIXED INTEGER LINEAR PROGRAMMING problem, using historical runtime measurement in the model. The results of this model are the optimal trip departure times for every trip at every stop. MIXED INTEGER LINEAR PROGRAMMING will be discussed in more detail in Section 2.3.

Timetable patterns are investigated in the PhD-thesis by Peeters [2003]. He models cyclic timetable patterns as a Periodic Event Scheduling Problem (PESP), where he uses an ILP-solver to optimize the timetable pattern for minimal vehicle cost and optimal connections between routes.

Optimization of a timetable with respect to vehicle cost and quality is a subject on which a lot of work is done. An overview of articles on this subject can be found in the introduction

of Bruno et al. [2009].

Amberg et al. [2011] discuss methods to increase robustness and punctuality while scheduling vehicles and personnel. The authors do this by redistributing the buffer times between trips, in order to get a more robust vehicle schedule and crew schedule at no extra cost. They use different models and compare the results of these models.

In his PhD-thesis, Huisman [2004] first discusses the sequential approach to vehicle and crew scheduling, where the vehicle schedule is optimized first, after which he uses this optimized vehicle schedule to determine the optimal crew schedules. Subsequently, he discusses the integrated planning of both stages, which leads to better solutions. For determining these schedules, he uses a combination of COLUMN GENERATION and LAGRANGIAN RELAXATION. These techniques are discussed in more details in Sections 2.4 and 2.5.

In van Kooten Niekerk [2010], we propose a way to estimate the crew cost in an early phase of planning, when determining the timetable pattern. As far as we know, the cost associated with the drivers can only be calculated when a full timetable is known. Because crew cost forms a large part of the total cost, we take this cost into account earlier in the planning process. Calculation of the crew cost is done by splitting the timetable pattern in cyclic vehicle tasks, where these cyclic vehicle tasks comply with crew-specific constraints, like breaks after a few hours of driving. For determining these timetable patterns, COLUMN GENERATION in combination with BRANCH-AND-PRICE is used.

1.5 Our contribution

In Chapter 2 we start with a brief description of some of the methods and background information that are used throughout this thesis. After these preliminaries, four new subjects of research are presented that form the core of this thesis.

Chapter 3 describes the improvement made to current trip runtime determination methods. We improve the method from Kroon et al. [2007] in order to take a combination of holding points and non-holding points into account, a concept that is investigated in van Oort [2011]. Furthermore, we also take passengers into account by considering average passenger travel and waiting time along with the traditional average delay and punctuality. Using a combination of holding points and non-holding points in the MIXED INTEGER LINEAR PROGRAMMING model of Kroon et al. [2007] makes the model intractable for medium to large sized instances, so we investigated other optimization techniques to solve the TRIP RUNTIME DETERMINATION PROBLEM.

In Chapter 4 we consider the problem of improving punctuality and robustness in vehicle scheduling. Traditionally, minimum trip layovers are applied to prevent propagation of disruptions; these minimum layovers typically have the same value for a lot of trips. To improve upon this, we first propose a method to calculate the punctuality of a vehicle schedule taking propagation of disruptions and historical runtime measurements into account. After this, we propose and evaluate several methods for vehicle scheduling using measured trip runtimes, where we minimize the average delay, taking into account the possible propagation of disruptions. We show that using our methods instead of using predefined minimum layovers gives a vehicle schedule with a better punctuality for the same cost.

In Chapter 5 we consider Electric Vehicles (EVs). Traditional vehicle schedule optimization often leads to infeasible schedules for EVs since the batteries are not big enough to enable a whole day of operation without recharging; therefore we take the properties of the batteries of the EVs into account at the time of vehicle scheduling. At the moment, in Europe the number of EVs in many networks is relatively small, so this can still be done manually. However, with the increasing popularity of these EVs, this will not be true anywhere in the near future. In Chapter 5, we propose two models and three solution techniques for the optimization of vehicle schedules for EVs. Each model has a different trade-off between problem size and quality of solution. The major part of this chapter has been published earlier as van Kooten Niekerk et al. [2017].

Chapter 6 discusses scheduling during the transition towards a cleaner and more sustainable public transport. During such a transition, we usually have to cope with a mix of older and newer, cleaner vehicles. Two different transitions are discussed. In Section 6.3 the situation where there is a set of vehicles available with different exhaust characteristics is discussed. By taking this into account during vehicle scheduling, we show that the air pollution in the whole area can be reduced significantly, along with an ever greater reduction in black spots in the city of Utrecht. In Section 6.4 we discuss vehicle scheduling where a mix of traditional and electric vehicles is available. We show that it is useful to take the characteristics of the different vehicles during the transition phase into account during vehicle scheduling.

In Chapter 7 we conclude this thesis with an overview of the conclusions in this thesis, along with open problems for further research.

Chapter 2

Preliminaries

In this chapter we will discuss some of the techniques and concepts that are frequently used throughout this thesis. No proofs will be given; for these we refer to the literature on the respective subjects.

2.1 DYNAMIC PROGRAMMING

In Chapter 3, Section 3.6 one of the methods used is DYNAMIC PROGRAMMING. DYNAMIC PROGRAMMING is a method for solving a complex problem by splitting it up into smaller sub-problems, solving these sub-problems and combining the results. DYNAMIC PROGRAMMING was first described in Bellman [1954]. The solution for every sub-problem is stored, so when it needs to be solved more than once, the stored solution is used. The idea is that when many sub-problems are equal, the optimization can be done much faster because for most sub-problems a stored solution exists and the number of sub-problems that has to be solved reduces dramatically.

We show this with an example for later reference, where we solve the SHORTEST PATH PROBLEM in a directed acyclic graph (DAG), which is widely used for many applications.

The SHORTEST PATH PROBLEM can be defined as follows: Given a directed acyclic graph $G(N, A)$ with nodes $n_i \in N$, arcs $a_{ij} \in A$ and costs c_{ij} associated to arcs a_{ij}, find the path from node n_{start} to $n_{destination}$ with minimal total cost of the arcs used in the path.

The shortest path in $G(N, A)$ can be found in the following way: Because $G(N, A)$ is a DAG, we can renumber the nodes in such a way that for every a_{ij} from n_i to n_j, index j is larger than index i. We want to find the shortest path from n_0 to n_{max}, where max is the number of nodes minus 1.

On the shortest path from n_0 to n_{max}, n_{max} has a predecessor, to which we refer as n_k. So the shortest path from n_0 to n_{max} consists of the shortest path from n_0 to n_k plus arc $a_{k,max}$. We do not know k, but we know that $n_k \in \{n_0, ..., n_{max-1}\}$, so when we calculate all shortest paths from n_0 to n_i with $i = 0, ..., max - 1$, we can easily determine the shortest path from n_0 to n_{max} by adding the arc $a_{i,max}$ and choosing the path with minimum length. We calculate all these shortest paths recursively using the following algorithm.

For every node n_i we keep track of the length of the shortest path from n_0, which we

denote by l_i and the number of the previous node of that shortest path, which we denote by p_i. In order to determine the shortest path, we execute the following algorithm:

```
1  l_0 ← 0 ;
2  l_1 ... l_max ← ∞ ;
3  for i ← 0 to max − 1 do
4  |  foreach Arc a_ij do
5  |  |  if l_i + c_ij < l_j then
6  |  |  |  l_j ← l_i + c_ij;
7  |  |  |  p_j ← i;
8  |  |  end
9  |  end
10 end
```

Algorithm 1: Algorithm to determine the shortest path in a DAG

In l_{max}, we find the cost of the shortest path from n_0 to n_{max}, and the shortest path itself can be reconstructed following the references in p_i from n_{max} back to n_0. The shortest path can be determined in linear time in the number of arcs.

In this example, the problem is determining the shortest path from n_0 to n_{max} and the sub-problems are finding the shortest paths from n_0 to a node n_i. By storing the optimization results in every node n_i, we do not have to enumerate all possible paths from n_0 to n_{max}, but we can use the values l_i instead of determining the shortest path from n_0 to n_i every time that n_i is in a possible route that must be evaluated.

In order to get the optimal result with DYNAMIC PROGRAMMING, there are two requirements for the complex problem: *optimal substructure* and *overlapping sub-problems*. For *optimal substructure*, it has to be proved that when the problem is split into sub-problems and the results are combined again, then the final result is optimal when the sub-problems are solved to optimality (Bellman's Principle of Optimality). In our example, we comply with this property because for all nodes n_i on a shortest path from n_0 to n_{max}, the shortest path from n_0 to n_i should be optimal, because otherwise we can replace the path from n_0 to n_i with the optimal path to make the shortest path from n_0 to n_{max} even shorter. *Overlapping sub-problems* are necessary for DYNAMIC PROGRAMMING to work efficiently; in our example we store the shortest paths from n_0 to n_i, which is part of many possible paths from n_0 to n_{max}.

A more extensive explanation, together with a proof of correctness of DYNAMIC PROGRAMMING for the case we discussed, can be found in Cormen et al. [2001], Chapter 16.

2.2 Convex optimization

In Chapter 3, Section 3.7 we apply continuous optimization, where we look for the optimum of a continuous function. When the function is convex, continuous optimization becomes a lot easier because of the properties of convex optimization. In this section we discuss some of

those properties. An extensive description of convex optimization can be found in Boyd and Vandenberghe [2004].

A function $f : \mathbb{R} \to \mathbb{R}$ is called a convex function when for every two points on the function curve, the connecting line lies above or on the function. We can express this formally and extend it to multi-dimensional functions as follows: A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex when it satisfies

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y) \tag{2.1}$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$.

The convexity of a function makes finding the minimum value of the function a lot easier. One key property of a convex function is that when one has a local optimum, it is the global optimum. For a function $f : \mathbb{R} \to \mathbb{R}$ this implies that on one side of the minimum, the function is descending and on the other side it is ascending. When one has two points a and b with $a < b$ on the function f on either side of the minimum, one can easily find the minimum of the function by reducing the interval. When we also assume that the derivative $f'(x)$ of $f(x)$ always exists in the given interval and is known, we may use the following simple procedure:

1. Take a point c halfway between a and b and calculate the derivative of c
2. If this derivative is positive, then set b to c, else set a to c
3. Repeat steps 1 and 2 until $b - a$ is smaller than the required tolerance.

For optimization of convex functions of which the derivative is not known, for multi-dimensional functions and for optimization with constraints, a lot of different algorithms have been described in literature. We refer to Boyd and Vandenberghe [2004] for an elaborate discussion of convex optimization.

2.3 LINEAR PROGRAMMING

LINEAR PROGRAMMING is a method for solving optimization problems with a linear objective and linear constraints. This class of optimization problems is widely used. In this thesis, we will encounter LINEAR PROGRAMMING in every chapter. A LINEAR PROGRAMMING problem is a special case of a convex optimization problem as discussed in Section 2.2 and is of the form:

$$\text{Maximize} \quad \sum_{j=1}^{m} c_j x_j \tag{2.2}$$

$$\text{Subject to} \quad \sum_{j=1}^{m} a_{ij} x_j \leq b_i \qquad \forall i = 1, ..., m \tag{2.3}$$

$$x_j \geq 0 \qquad \forall j = 1, ..., n \tag{2.4}$$

where x_j are the variables to be determined, c_j is the cost associated with variable x_j and a_{ij} and b_i define the linear constraints.

The most famous solution algorithm for solving linear optimization problems is the *Simplex method* developed by George Dantzig in 1947. The result of the Simplex method is the optimal value for $\sum_j c_j x_j$ together with the values for x_j where this optimum is achieved. Furthermore, for every constraint i, the *shadow price* μ_i is calculated. This shadow price indicates by what extent the optimal value will change when the value b_i for constraint i is changed. When these shadow prices are multiplied by the values a_{ij} in the column belonging to variable x_j we get the *reduced cost* of this variable; this value is equal to $c_j - \sum_i a_{ij}\mu_i$. The reduced cost represents the influence which this column has on the optimal value. When the reduced cost is positive, the optimal value will increase when the column is included, while a column with a non-positive reduced cost will not have any influence on the value of the optimum.

In this thesis, we also use a special case of LINEAR PROGRAMMING, where one or more variables x_j are required to be integer. This is called MIXED INTEGER LINEAR PROGRAMMING, and the case where all variables x_j are required to be integer is called INTEGER LINEAR PROGRAMMING. In order to solve MIXED INTEGER LINEAR PROGRAMMING and INTEGER LINEAR PROGRAMMING problems, two techniques are widely used: Cutting-Planes and Branch-and-Bound. Both methods start with solving the LP-relaxation with the Simplex method; the LP-relaxation is obtained by relaxing the integrality constraints.

The *Cutting-Plane method* was first described in Gomory [1958]. This method starts with solving the LP-relaxation. When there are non-integer variables which should be integer, there should be a constraint that separates this non-integer solution from the integer solution space. The Cutting-Plane method determines such a constraint and adds it to the original problem. Then the Simplex method is run again. This process is repeated until all integrality constraints are obeyed or the collection of cutting planes is exhausted.

The *Branch-and-Bound technique*, first described in Land and Doig [1960], uses a different approach. When a non-integer solution occurs, one integral variable that currently has a fractional value is chosen. Then two copies of this problem are made, with both an additional constraint: one where the variable should be at least its current value rounded up, and one where the variable should be at most its current value rounded down. One of these two sub-problems will contain the optimal value. Both sub-problems are solved, and the best solution of both is the optimal solution. If a non-integer solution appears in a sub-problem, the Branch-and-Bound technique is applied recursively.

2.4 COLUMN GENERATION

For many real-life problems, formulating it as a LINEAR PROGRAMMING problem will lead to a problem with very many variables. Examples of these problems are vehicle scheduling and crew scheduling problems, where every variable represents a possible vehicle task or duty, and the constraints ensure that every trip is driven at least once. Often it is intractable to use or even generate all variables. In this case we may use COLUMN GENERATION. In this thesis, COLUMN GENERATION is used in Chapter 5.

Solving a problem using COLUMN GENERATION works as follows. The problem is split in two parts. The first part is the *Restricted Master Problem* (RMP), which is a LINEAR

PROGRAMMING problem with a subset of the variables of the full problem. This RMP is solved. From this solution, we use the shadow price for every constraint. The second part is the pricing problem. Here we want to determine a new variable that will improve the current solution of the RMP, together with its column. The reduced cost of such a column represents its influence on the optimal value of the RMP. If the reduced cost is positive, the optimal value of the RMP is likely to increase. We look for a variable and a column with positive reduced cost by solving the pricing problem. This is an optimization problem in which we find the variable with maximum reduced cost; it is crucial that the pricing problem can be solved efficiently. Subsequently we add this variable to the RMP, and the new RMP is solved. This process is repeated until the pricing problem is not able to generate a column with positive reduced cost any further; in this case, we have solved the LP-relaxation to optimality.

COLUMN GENERATION is applicable to LINEAR PROGRAMMING problems, but many problems require an integral solution. One method to acquire an integral solution is to use *Branch-and-Price*, which is a hybrid of COLUMN GENERATION and Branch-and-Bound. This method is described in Barnhart et al. [1998]. Branch-and-Price will give an optimal solution, but it may take very long. We can also use a heuristic. One of these heuristics is called *Truncated Column Generation*, which we will use in Section 5.5.3 on page 96. In this heuristic we use COLUMN GENERATION to obtain a (possibly fractional) solution for our problem. If this solution is integral, then we are finished. Otherwise we choose one or more variables that are integral or almost integral, fix the value of the variable to the nearest integer and remove the variable and column from the problem, which will reduce in size. Then we solve this smaller problem with COLUMN GENERATION and repeat the process until the solution is integral.

More in-depth information about COLUMN GENERATION can be found in Desaulniers et al. [2005].

2.5 LAGRANGIAN RELAXATION

LAGRANGIAN RELAXATION was first described by Geoffrion [1974]; it is a technique to approximate the result of a difficult optimization problem by relaxing constraints. The method removes constraints from the problem and adds a cost for violating the constraint to the objective function. Usually, the relaxed problem can be solved more easily than the original problem.

For ease of exposition, we assume that we can formulate the problem at hand as an INTEGER LINEAR PROGRAMMING. Hence, we get

$$\begin{aligned} \text{Maximize} \quad & c^T x && (2.5)\\ \text{Subject to} \quad & Ax \leq b && (2.6)\\ & x \geq 0 && (2.7) \end{aligned}$$

where $x \geq 0, x \in \mathbb{N}^n$, $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$.

We split the constraints in $Ax \leq b$ in easy-to-handle constraints and difficult constraints. For example: when we have a matching problem with additional constraints, the matching

problem without constraints results in a Totally Unimodular Matrix (TUM). This always results in an integral solution when solving the relaxed LP. These constraints are considered easy-to handle, while adding the additional constraints may lead to a matrix that is not TUM anymore, and hence the constraints are considered difficult. Suppose the easy-to-handle constraints are put in $A_1x \leq b_1$ and the difficult ones in $A_2x \leq b_2$. This gives:

$$\begin{aligned} \text{Maximize} \quad & c^T x && (2.8)\\ \text{Subject to} \quad & A_1 x \leq b_1 && (2.9)\\ & A_2 x \leq b_2 && (2.10)\\ & x \geq 0 && (2.11) \end{aligned}$$

where $A_1 \in \mathbb{R}^{m_1,n}$, $b_1 \in \mathbb{R}^{m_1}$, $A_2 \in \mathbb{R}^{m_2,n}$, $b_2 \in \mathbb{R}^{m_2}$ and $m_1 + m_2 = m$.

We move the constraints A_2 into the objective, introducing non-negative weights $\lambda \in \mathbb{R}_{m_2}$ for these constraints:

$$\begin{aligned} \text{Maximize} \quad & c^T x + \lambda^T (b_2 - A_2 x) && (2.12)\\ \text{Subject to} \quad & A_1 x \leq b_1 && (2.13)\\ & x \geq 0 && (2.14) \end{aligned}$$

The vector λ is called the vector of *Lagrangian Multipliers*. Because $\lambda \geq 0$, for every given λ the optimum for the relaxed problem is never less than the optimum for the original problem, so LAGRANGIAN RELAXATION gives an upper bound of the original problem. We can minimize this upper bound by looking for the $\lambda \geq 0$ where the optimum for the relaxed problem is minimal. This problem is called the *Lagrangian Dual Problem* and gives an upper bound for the original problem that dominates the bound obtained by solving the LP-relaxation; Geoffrion [1974] has shown that these bounds coincide if the integrality constraints are redundant in the Lagrangian relaxation.

In this thesis, we will use the property that the optimal value of λ of the Lagrangian dual problem can be used as dual costs in COLUMN GENERATION in Chapter 5.

More detailed information on LAGRANGIAN RELAXATION in combination with COLUMN GENERATION can be found in Chapter 9 of Desaulniers et al. [2005].

2.6 NP-hard problems

For the optimization problems we encounter in later chapters, we want solutions that are efficient and scalable. This combination of features is typically guaranteed in solutions whose computation time is bounded by a low-degree polynomial (as in the SHORTEST PATH PROBLEM). However, many optimization problems seem intrinsically not to be solvable in polynomial time. A classical perspective on this issue is provided by the theory of 'hard problems' (Garey and Johnson [1979]). We recall some of the pertinent notions.

We first distinguish between the *decision version* of an optimization problem ('does a desired solution exist') and the *construction version* of the problem ('determine a desired

solution'). Let **P** be the class of decision problems solvable in deterministic polynomial time, and **NP** the class of decision problems solvable in *nondeterministic* polynomial time. For many familiar optimization problems, such as TRAVELING SALESMAN PROBLEM or INTEGER LINEAR PROGRAMMING, the decision version is known to be in **NP** but not known to be in **P**. The general question whether **P** is equal **NP** or not, is a well known and important *open problem* in computational complexity.

An optimization problem $\mathcal{A}$ is called *NP-hard*, if every decision problem $\mathcal{B} \in$ **NP** can be polynomially reduced to $\mathcal{A}$, i.e. can be solved by a deterministic polynomial time algorithm that can solve instances of $\mathcal{A}$ in 'one' step. Clearly, if an NP-hard problem were polynomially solvable, then the **P** =**NP** problem would be solved. As many of the planning and scheduling problems in Chapters 3-6 are known to be NP-hard in general, exact approaches that solve them in polynomial time are not known and generally assumed not to exist (unless **P** =**NP**).

To obtain solutions for NP-hard problems that still work well in practice, we either have to defy their NP-hardness and use techniques like those in Sections 2.1-2.5 or design close approximations or heuristics that are effective. For further details of the approaches in this field, we refer to Garey and Johnson [1979], Hochbaum [1997] and Williamson and Shmoys [2011].

2.7 Set Partitioning and Set Covering

In many planning and optimization problems, we have a set of elements which have to be covered by exactly one or at least one subset. For example: a municipality has to make sure that every street is served by at least one garbage truck every week. Or a postal service has to divide the letters to be delivered between the postmen. Besides picking up all the garbage and delivering the mail, the company typically wants to reduce cost by optimizing the assignment of streets or letters. In public transport, every trip has to be driven by one vehicle.

The minimization version of the SET PARTITIONING PROBLEM can be formulated as: Given a finite set S and a finite family F of subsets of S, with cost c_f associated with each $f \in F$, find a minimum cost subfamily $F'$of F such that the subsets in F' form a partition of S.

This problem can be formulated as an INTEGER LINEAR PROGRAMMING problem. For this, we use a binary variable x_f for every $f \in F$. x_f is set to 1 if f is part of F' and 0 otherwise. We use a_{if} to indicate whether subset $f \in F$ contains element $i \in S$. The SET PARTITIONING PROBLEM can be stated as:

$$\text{Minimize} \quad \sum_{f \in F} c_f x_f \tag{2.15}$$

$$\text{Subject to} \quad \sum_{f \in F} a_{if} x_f = 1 \qquad \forall i \in S \tag{2.16}$$

$$x_f \in \{0, 1\} \qquad \forall f \in F \tag{2.17}$$

The SET COVERING PROBLEM is a relaxation of the SET PARTITIONING PROBLEM. Instead of the requirement that F' is a partition of S, we require that every element $i \in S$ is in at least one subset $f \in F'$. In Equation 2.16, we use '$\geq$' instead of '$=$'.

SET PARTITIONING PROBLEMs occur often in vehicle and crew scheduling problems, where the elements i are the trips or activities to be scheduled, and the subsets f represent the set of trips that can be driven by one vehicle or one driver. SET PARTITIONING PROBLEM and SET COVERING PROBLEM are examples of NP-hard problems, see Garey and Johnson [1979]. This means that these problems are not known to be solvable in polynomial time; calculation time may become prohibitive with large instances (cf. Section 2.6).

In this thesis, SET PARTITIONING PROBLEMs are used in Chapters 4, 5 and 6.

2.8 The SHORTEST PATH PROBLEM with additional constraints

When using COLUMN GENERATION in a vehicle scheduling context, as we do in Chapter 5, the pricing problem usually contains a SHORTEST PATH PROBLEM on a graph that represents all trips and links between them. Because these trips have fixed times, and because we only can go forward in time, this graph is a DAG (directed acyclic graph).

Finding the shortest path in a DAG can be done fast by using DYNAMIC PROGRAMMING, as described earlier in Section 2.1.

If there are additional constraints on a path through the graph, we may have to use an adaptation of the algorithm described in Section 2.1. Instead of storing only the minimum cost and a reference to the previous node in the shortest path, we store additional information about that path. For example in Section 5.5.3, we have to find a shortest path through the graph, where the battery of an electric vehicle does not run empty when driving along the shortest path. For this we need to keep track of the state of charge of the battery. One method to take this (and possibly other things) into account is the label-correcting algorithm, which is described for the ELEMENTARY SHORTEST PATH PROBLEMS WITH RESOURCE CONSTRAINTS in Feillet et al. [2004], which is used and shown to work well for scheduling electric vehicles in Huang and Li [2016]. We will explain this algorithm briefly.

The algorithm described in Huang and Li [2016] is based on the algorithm described in Section 2.1. In this last algorithm, the minimum distance from node n_0 is stored in l_i and the previous node in this shortest path is stored in p_i. When we have an extra constraint, in this case because we have to keep track of the electric energy that is still in the battery, we have to adapt this algorithm. Instead of using only information on the shortest path at every node (l_i and p_i), we also have to keep track of the state of charge of the battery. Because there could be multiple paths to this node with different states of charge of the battery, we keep track of the shortest paths for every possible value of state of charge at this node, and we store this information in a label, which is attached to a node. If a label is dominated by another label, it is not kept. Every node could have more than one label. We scan the labels corresponding to the same node to remove dominated labels: a label is dominated if there exists another one with equal or lower length and an equal or higher amount of electricity left in the battery. When traversing all nodes, we follow the outgoing arcs in combination with all labels. When there is a valid path, we update the labels. When there is no label for the current state of charge, we add the label. Otherwise we update the label when the new

path is shorter than the length stored in the current label on the node. At the end, we look for the label with minimum length on the last node and track back the labels in order to get the shortest path. Huang and Li [2016] show that the RESTRICTED SHORTEST PATH PROBLEM they are solving is NP-hard.

Chapter 3

Determination of planned trip runtimes

Most, if not all, public transport systems in the world have a timetable. These timetables are published on the internet and in leaflets, and stop posters are published at the stops, so that passengers know at what time they can expect a vehicle. In reality a vehicle never exactly arrives according to a timetable, because of various disturbing factors like traffic conditions. Nonetheless, the planned departure times at a stop should be close to the real departure times, in order to improve punctuality, reduce waiting time and avoid discomfort for the passenger. In this chapter we consider the problem of finding the optimal timetable for trips along a given route. We will first discuss the problem in detail, after which we will develop methods to determine an optimal timetable for one route.

In most public transport organizations (PTOs) in the Netherlands, the arrival and departure times of every trip at every stop are measured. These measurements are used for the determination of the appropriate planned departure times that are published at the stop and in timetables. Currently, in practice, rules of thumb are used, such as a given percentile of the measured driving or departure times. We will propose and evaluate several methods to determine the set of optimal planned departure times, where we are able to optimize on different objectives, such as average delay and average passenger waiting time. The problem of determining an optimal timetable for a route, given the measured departure times, is called the TRIP RUNTIME DETERMINATION PROBLEM.

In a trip route, we may have one or multiple holding points. These are stops where vehicles that are too early, wait until their planned departure time. Through the waiting time at a holding point we can influence the passing times at stops after the holding point. When a vehicle arrives too early at a holding point, the vehicle and the passengers in it have to wait until the planned departure time. When a vehicle arrives later than the planned time, the passengers at this stop have to wait for the delayed vehicle. Both cases are undesirable and both lead to longer travel times for the passengers; because of the variation in the driving times, we cannot avoid both, but we can reduce their negative impact by finding good planned departure times.

Observe that trip runtimes are influenced by road congestion and passenger numbers and change during the day. Typically, a vehicle needs more time in rush hours and less time in the evening. We choose to model the driving times by a set of measurements, which implicitly gives us an empirical probability distribution. When we would use the same trip time model

for the whole day, it would not fit very well. Instead of this, we use different distributions and hence different timetables for different time periods. We call these time periods for timetables *runtime groups*.

Our goal in this chapter is to determine several new methods to solve the TRIP RUNTIME DETERMINATION PROBLEM, taking into account all the above considerations. In all methods we propose, we will explicitly take into account that not all stops are holding points, contrary to what is done in the known methods from literature and in the methods that are widely used in practice. We will discuss several different optimization objectives; besides the widely used objectives like punctuality and average delay we will also use passenger-related objectives, like average waiting time. We will consider these objectives separately and also combine them in a weighted sum. We will also perform computational experiments to compare our new methods to a few well-known methods from the literature. Furthermore, we will evaluate the effect of splitting a day into two or more runtime groups and solving the TRIP RUNTIME DETERMINATION PROBLEM for each part separately.

This chapter begins with an overview of literature on the TRIP RUNTIME DETERMINATION PROBLEM in Section 3.1. We perform a thorough analysis of actual trip runtime measurements in Section 3.2. Information from this analysis will be used to formulate the objectives in Section 3.3. In the next four sections we will study four different methods for solving the TRIP RUNTIME DETERMINATION PROBLEM. In Section 3.4 the percentile method is discussed; this method is currently used in most public transport organizations.

In Section 3.5, we will discuss a method based on INTEGER LINEAR PROGRAMMING described in Kroon et al. [2007] which is an extension of a method that is used by the Dutch railroad company NS (Nederlandse Spoorwegen, Netherlands Railways). In Kroon et al. [2007], the vehicle waits at every stop until its planned departure time, whereas we do not require this for every stop, which complicates the problem. Our method based on INTEGER LINEAR PROGRAMMING finds optimal results, but turns out to be too slow for real-life instances. In Section 3.6, we investigate the use of a DYNAMIC PROGRAMMING heuristic for solving the TRIP RUNTIME DETERMINATION PROBLEM. This turns out to give better results than the percentile method, with a reasonable computation time.

When we allow the planned departure times to be non-integer, we can also use continuous optimization, which is discussed in Section 3.7. With our definition of the problem as an unconstrained continuous optimization problem, we can solve large instances very fast with a very good result. All the methods mentioned can be divided into two parts: methods that require planned times to be integral and methods that do not. For the latter, integral times can be obtained by rounding. Integral times are required for INTEGER LINEAR PROGRAMMING and the DYNAMIC PROGRAMMING heuristic, whereas the percentile method and continuous optimization require continuous times. We evaluate both groups and compare the methods in each group in Section 3.8.

We will show that our methods perform significantly better than the currently used methods. Depending on the focus in the optimization, the number of tardy departures is halved or the average waiting time at the stop for the passenger is halved.

Another method to improve timetable quality is by solving the TRIP RUNTIME DETERMINATION PROBLEM for parts of the day separately, instead of one instance of the TRIP RUNTIME DETERMINATION PROBLEM for the whole day. These are the runtime groups

we mentioned earlier, they are widely used in combination with the percentile method (see van Oort [2011]). In Section 3.9 we investigate the use and practicality of runtime groups in combination with our new methods. We end this chapter with a conclusion in Section 3.10.

3.1 Literature overview

In van Oort [2011], the author discusses trip runtime determination for buses using the Percentile Method. In this method, the planned passing times at stops are determined by taking a percentile value of all measured passing times while driving a previously planned timetable. The author uses three fixed percentile-values: one for the total trip duration, one for the passing times at holding points during a trip, and one for the passing times at all other intermediate stops. The total planned trip duration is fixed to a percentile value of all measured trip durations of 85%. If there are no holding points, then using a percentile value of 35% performs best, when looking at the minimization of the total travel duration. Here passengers who miss the bus have to wait for the next bus. Vehicle holding is evaluated as an instrument to increase punctuality. For the holding points, a percentile value of the departure time in the range between 50% and 65% is found. The author also discusses the benefits of using runtime groups. His conclusion is that it is very useful to apply them, but that it is generally not useful to apply more than 4 runtime groups in a day.

Kroon et al. [2007] and Vromans [2005] considered the TRIP RUNTIME DETERMINATION PROBLEM for NS. For train planning in the Netherlands, the runtime on a track is determined by calculating the technically minimal running time. On top of this, a runtime supplement time of 7% is added to the technically minimal running time for the whole route between two main stations. The TRIP RUNTIME DETERMINATION PROBLEM here consists of distributing the runtime supplement over the stations on the route in order to minimize the average delay of the train. Trains will always wait for the planned departure time, so all stations are holding points. In Kroon et al. [2007] and Vromans [2005], some methods to distribute runtime supplements over a train trip based on INTEGER LINEAR PROGRAMMING are discussed. The authors show that by optimizing the runtime supplement distribution over the trip, a gain in the number of in-time station departures can be achieved of 30% in comparison to the current practice where this runtime supplement is distributed evenly over the trip.

Piester and Thorhauge [2010] translate passenger travel and waiting time into cost. The authors optimize the total cost, which consists of real vehicle and crew cost and travel and waiting time cost, by changing the total runtime supplement in trips. They distribute this runtime supplement evenly over the stops and perform simulations in order to determine the passenger travel and waiting time. More runtime supplement may lead to more punctuality, but also to more cost for extra vehicles and crew. Less runtime supplement will lead to less punctuality and passenger dissatisfaction. For their instance of the Sydbanen in Denmark, they found an optimum when using a runtime supplement of 6 to 8 percent of the trip duration.

The planned trip departure times themselves also influence the travels passengers make. They may have to wait shorter or longer at their departure stop, and their travel itself may take longer or shorter. In van Hagen [2011], the author investigates waiting time perception at

train stations and studies the effect of several ways to alleviate waiting time. We will not take waiting time alleviation into account, but we will use his result that waiting time at a platform is perceived as lasting about twice as long as travel time.

3.2 Analysis of trip runtime measurement data

In order to be able to develop a good method to solve the TRIP RUNTIME DETERMINATION PROBLEM, we first study the properties of trip runtimes in general. In this section, we will study the properties of trips in real life and formulate observations that we apply to our models in the following sections and in Chapter 4.

For the trip time determination, we assume that every bus leaves exactly on time at the start of the trip. For each stop, it would be desirable that a bus driver waits for the planned departure time when he/she is early. However, in reality this is not always possible. One possible reason is that a waiting bus blocks the road for other traffic. In the analysis of current run times, we see that it is common to wait for the planned departure time on main stops, like a train station or an intersection with other routes. In the optimization of the punctuality, we will use mainly these points as holding points. This leads to our first observation:

Observation 1. *A bus driver tends to wait for the planned departure time only at main stops.*

The runtime measurements we use are provided by Qbuzz, a public transport company in the Netherlands. They were acquired from the Automatic Vehicle Location-system (AVL-system). The bus sends a signal with its GPS-coordinate to the central computer when it arrives at a stop, when it leaves from a stop and when it is more than 30 seconds since the last signal was sent. In this way, a measurement is made at least every 30 seconds. In this section we will have a closer look at these measurements. We will look at the total trip duration during a day, at the runtimes during a trip, and we will look at the correlation between different runtimes on different parts of the trip.

For Figures 3.1, 3.2 and 3.3 in this section we have used trip measurements on route 6 in Groningen, the Netherlands, during 10 weekdays in October 2012. We use this route as a typical case, as our experience is that it is representative for the majority of other public transport routes. In Figure 3.1, we have put red dots for every trip to see the relation between planned departure time and measured trip duration. Note that the hours after midnight are not reset to zero, but will continue counting. The blue lines show the planned trip duration.

What we see in Figure 3.1 is that the total measured trip duration for each trip before 18:00 varies between approximately 34 and 50 minutes and that the trips after 18:00 tend to be shorter than the ones before 18:00. The planned trip duration changes at 7:00 and at 18:00; these are the boundaries of the three runtime groups that are used. In Section 3.9 we will deal with the variation of trip duration during the day.

For the same data as used in Figure 3.1, for the time frame from 7:29 to 17:51, we have made a time/stop graph (Figure 3.2) with a black line for every measured trip. The green line shows the planned runtime and the red brackets show the range of stop passing times that are considered as on time, in this case not too early and not more than three minutes delayed.

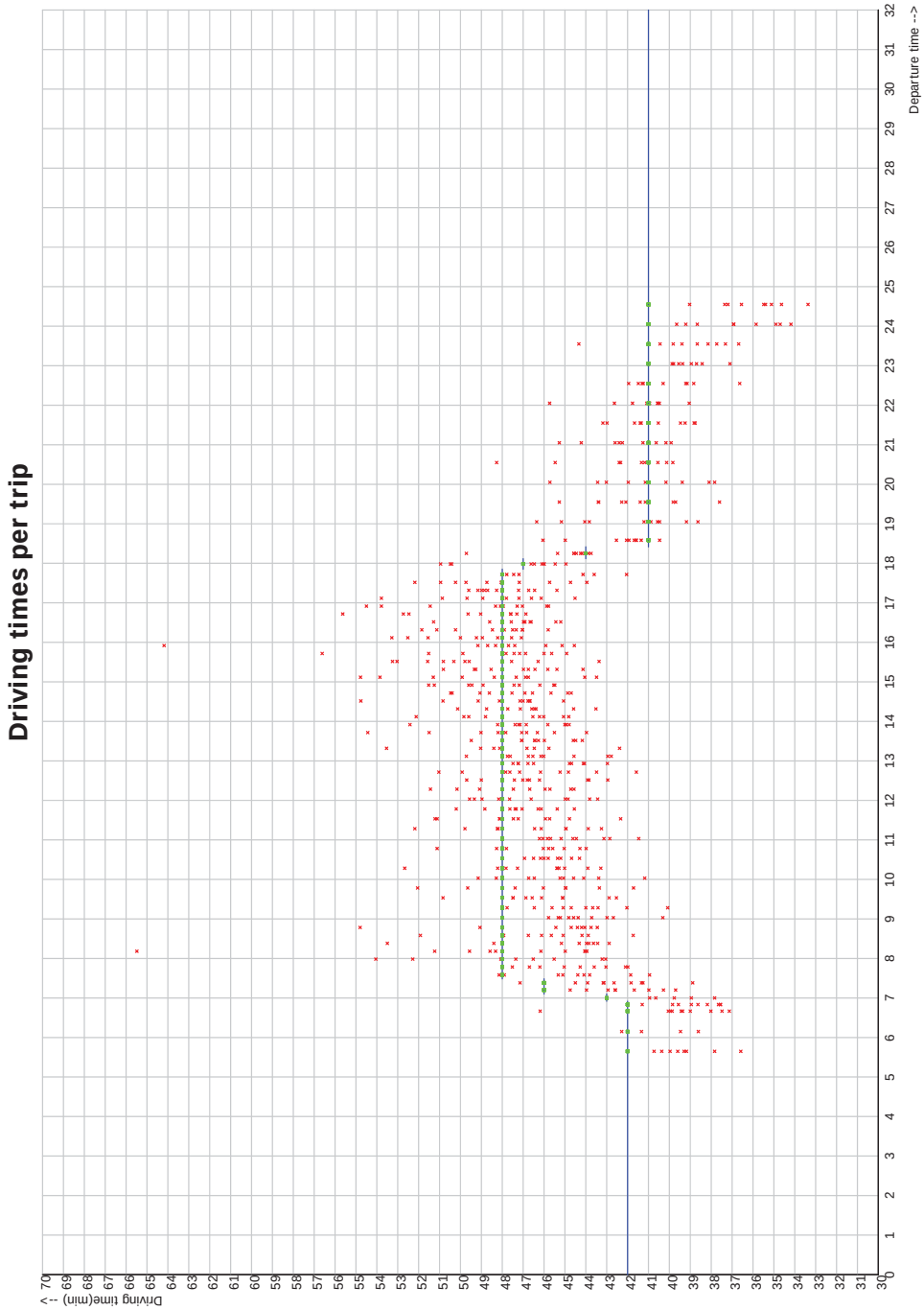


Figure 3.1: Driving time per trip on route 6. On the horizontal axis, we put the planned departure time of a trip, where the time will not be reset to zero at midnight, but will count further. On the vertical axis we put the trip duration. Every red dot denotes the actual duration of a trip, the blue line and the green dots denote the planned trip durations. The planned trip duration changes at 7:00 and at 18:00, these are the boundaries of the three runtime groups that are used.

Black lines that are not crossing the red brackets are not considered on time, they are either too early or too late.

In this figure, we see that most trips start on time. A few trips start more than 2 minutes early, but are mostly on time at the next stop. Thorough analysis of this showed that this was caused by a measurement error caused by the AVL-system that was used. Furthermore, we see that the black lines tend to disperse more during the trip, the variation increases. There are two stops with something remarkable: At 'Centraal Station', we see that the variation after this stop is smaller than just before the stop. This could imply that drivers wait for the planned departure time at this stop. The second remarkable thing happens between stops 'Grote Markt' and 'UMCG Noord'. Here the variation becomes much larger. This turned out to be caused by malfunctioning traffic lights on this part of the route. What we learn from this figure, is that drivers do not wait until their planned departure time unless it is a holding point (like 'Centraal Station'). And even there they do not always wait for the planned departure time.

A third figure, Figure 3.3, is used to analyze the covariance between measured runtimes on different parts of the earlier used route. A yellow rectangle indicates no correlation between the two trip segments, a green one reflects a positive correlation and a red one a negative correlation.

It is obvious that the diagonal itself is all green, but also adjacent to this diagonal green rectangles are found. This means that runtimes on adjacent segments have a positive correlation. When a part of a route is busy, we may expect that it is also busy a few hundred meters from there. What we notice is the vertical red line at 'Centraal Station'. As we noticed in the previous figure, drivers are probably waiting to leave at the right time. In this graph we see a negative correlation, which means that when a driver is fast on a part before the station, he has to wait longer at the station. So this graph confirms that drivers wait. We see the same effect at 'P+R Kardinge'. From this Figure 3.3, we observe that runtimes on nearby trip segments are correlated to each other and that there are a lot of other correlations. So we may not assume that runtimes are independent from each other. When optimizing runtimes, we can not use separate measurements between each pair of stops, but we will have to use measurements of complete trips. When looking at other routes, we see similar properties. This leads to the following observation:

Observation 2. *Actual runtimes in parts of a trip are often correlated. We cannot assume that they are independent.*

Because we saw that there is a correlation between the deviation from the planned driving times on some parts of a trip, we will use the measurements from whole trips instead of generating random durations based on statistics of runtimes of each trip segment. In the analysis, we saw that at some points, drivers tend to wait for the planned departure time when they arrive early. This dwelling time during which a driver waits at the stop until the planned departure time should be excluded from the trip runtime measurements because they are not real driving times, but are caused by planned departure times. We want to exclude the influence of an earlier trip runtime determination in our problem.

We looked at all the times the driver is standing still at a stop, the *dwelling times*. We look for a correlation between the arrival punctuality (the difference between the measured

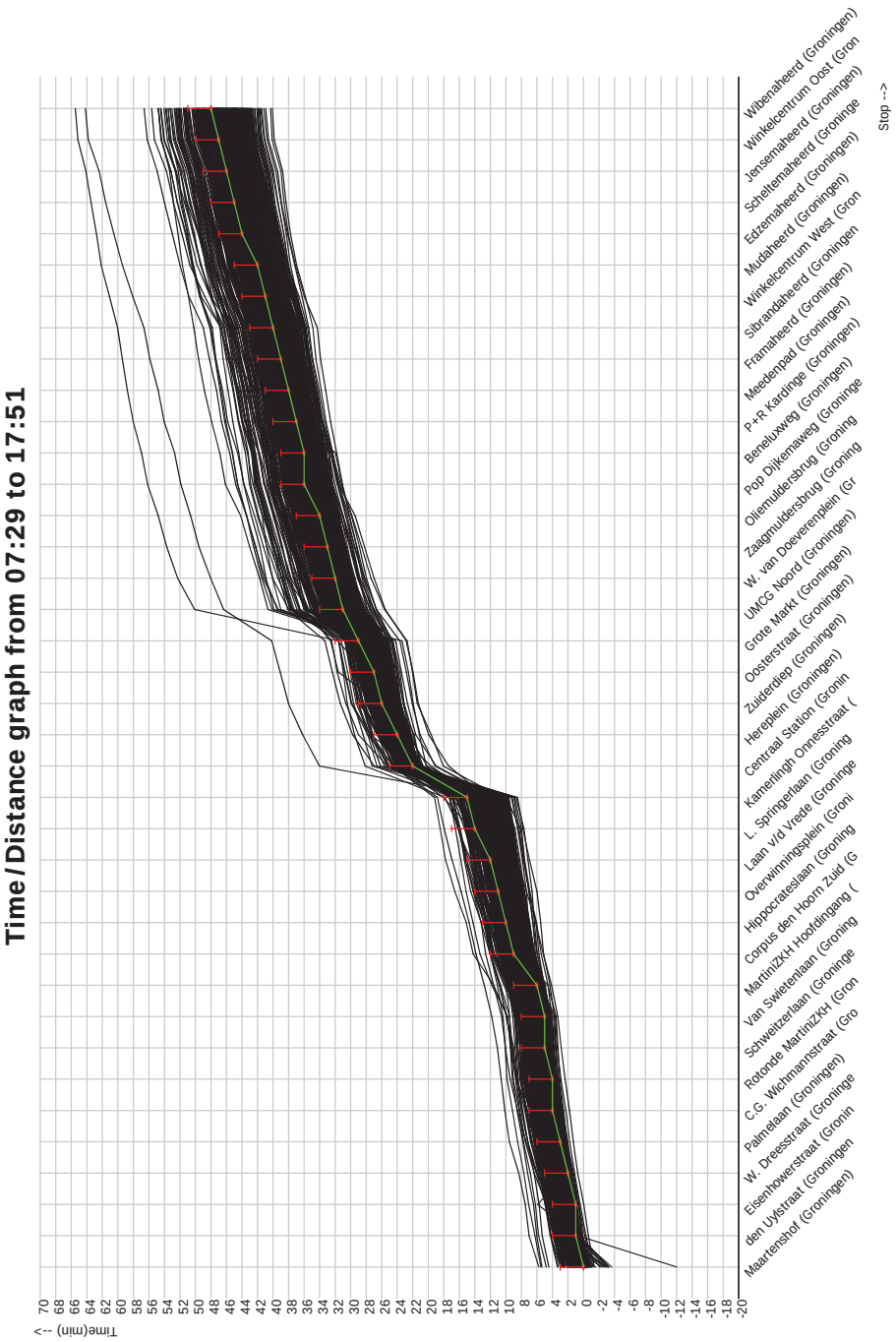


Figure 3.2: Time/stop graph of route between 7:29 and 17:51. On the horizontal axis, the stops are shown. On the vertical axis the duration between the planned departure time at the first stop and the actual time at this stop. The green line denotes the planned times, while the red brackets indicate which passing time at a stop is considered on time.

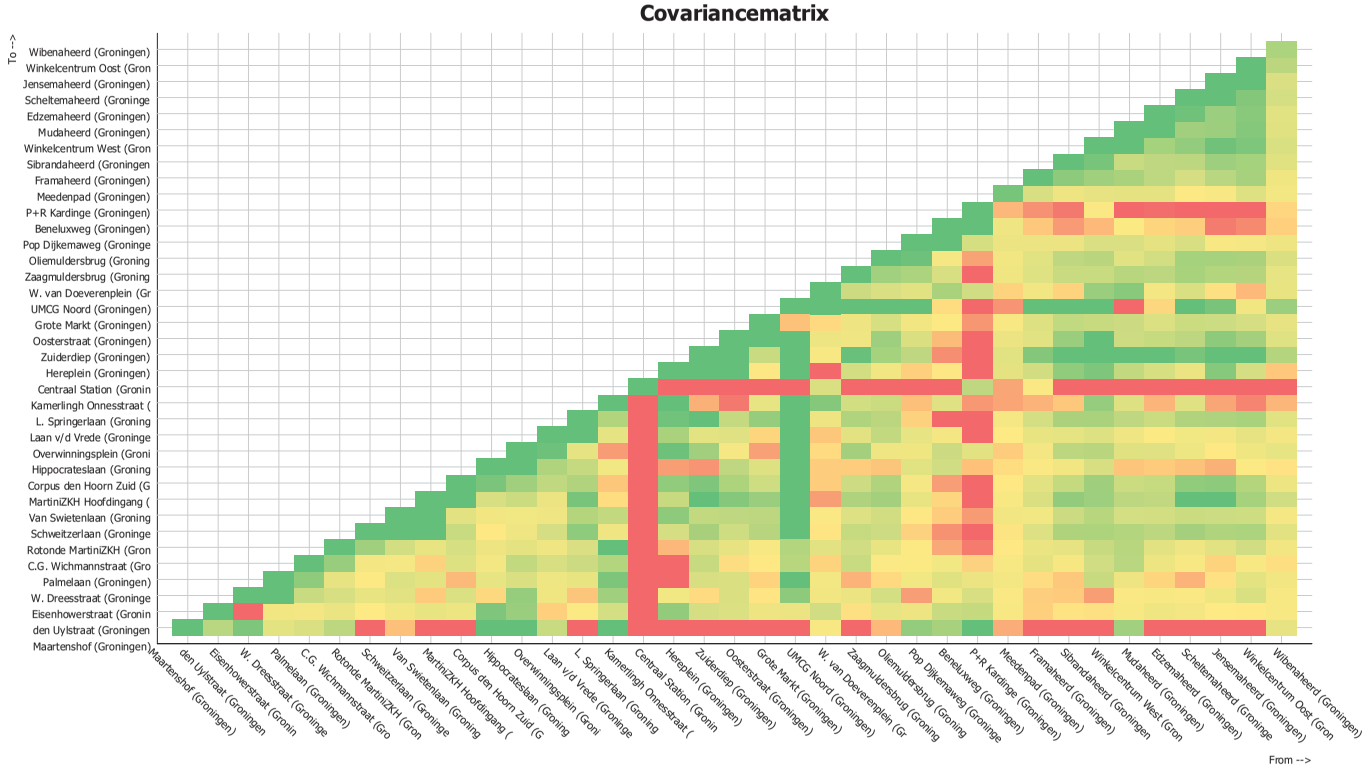


Figure 3.3: Covariance graph of route 6. On the horizontal and vertical axes, the stops are placed. Every colored rectangle denotes the covariance between two parts of the trip. A green rectangle represents a full positive correlation between two parts of a trip, a red rectangle represents a negative correlation.

and planned arrival) and the dwelling time. In order to put this data in a graph, we grouped the data in arrival punctuality intervals of 10 seconds. For every group we draw a box in the graph in Figure 3.4. The black line in the center of the box between the red part and the blue part is the 50th percentile, the top of the blue box shows the 70th percentile while the top of the vertical blue line is equal to the 90th percentile. Similarly the bottom of the red box is the 30th percentile and the bottom of the vertical red line represents the 10th percentile.

When we look at Figure 3.4, we see that the dwelling time at a stop is less than 45 seconds in 90 percent of the measurements when the driver does not arrive too early. We also see that the driver tends to wait longer when he/she arrives too early. So we can remove the extra dwelling time that is spent on waiting for the planned time by maximizing the dwelling time to 45 seconds. This leads to the following observation:

Observation 3. *When a bus dwells at a stop for more than 45 seconds, the driver is waiting for its planned departure time. Excluding the influence of an earlier trip runtime determination can be done by maximizing the dwelling time at a stop to 45 seconds*

Prior to all optimization, we remove any dwelling time in excess of 45 seconds.

3.3 Definition of the TRIP RUNTIME DETERMINATION PROBLEM

In this section, all prerequisites for solving the TRIP RUNTIME DETERMINATION PROBLEM are defined. In Section 3.3.1, we define the parameters and variables that are used in the constraints in Section 3.3.2. Some possible objectives are described in Section 3.3.3, which are translated into a cost function in Section 3.3.4.

3.3.1 Notation used in the TRIP RUNTIME DETERMINATION PROBLEM

For the TRIP RUNTIME DETERMINATION PROBLEM, we use measurements for one route in one direction. We model the trip run times as follows. We consider three sets of times. Set $\mathcal{M}$ contains the measured departure time for every trip. Set $\mathcal{P}$ contains the planned departure times to be determined by the TRIP RUNTIME DETERMINATION PROBLEM. Finally, set $\mathcal{C}$ consists of the runtimes for every trip, resulting from the departure times from set $\mathcal{M}$ and taking the planned departure times from set $\mathcal{P}$ into account. Set $\mathcal{C}$ also contains times when the vehicle is ready to leave, when this time is earlier than the planned departure time, the vehicle has to wait. All times are relative to the planned time at the first stop, which is set to 0. Note that set $\mathcal{P}$ only contains one set of times, while sets $\mathcal{M}$ and $\mathcal{C}$ contains one set of times for every measured trip. For every measured trip, we have also counted the number of passengers entering and leaving the bus at every stop.

We use the following parameters:

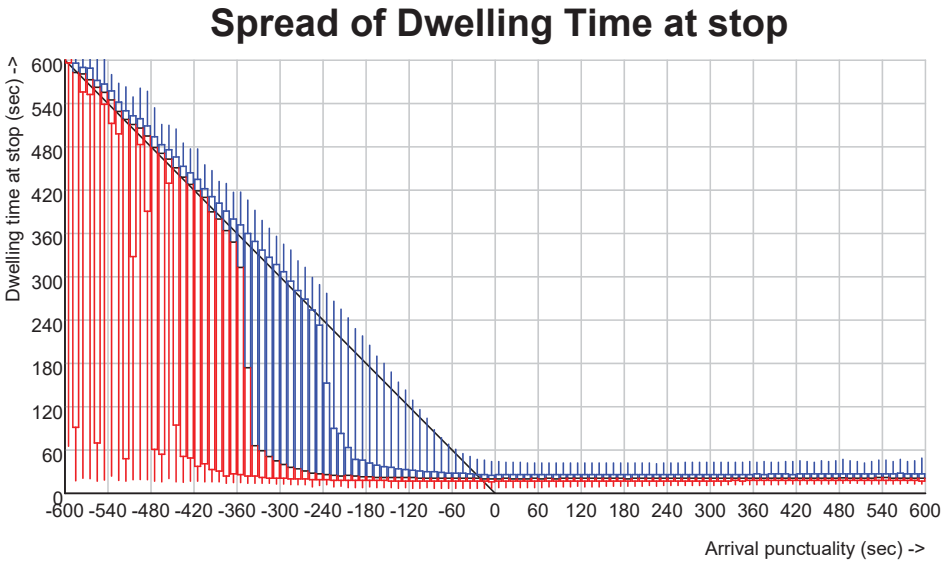


Figure 3.4: Graph of the dwelling time distribution at stops in relation to the arrival punctuality. On the horizontal axis, the punctuality at arrival is shown, on the vertical axis the actual dwelling time is shown. For every interval of 10 seconds, a vertical red or blue box is drawn. The black line in the center of the box between the red part and the blue part is the 50th percentile, the top of the blue box shows the 70th percentile while the top of the vertical blue line is equal to the 90th percentile. Similarly the bottom of the red box is the 30th percentile and the bottom of the vertical red line represents the 10th percentile.

n_{trip}	:	Number of measured trips
n_{stop}	:	Number of stops in trip, $n_{stop} > 1$
n_{hp}	:	Number of holding points
S_{stop}	:	Collection of all stops on the route of the trip
S_{hp}	:	Collection of stops that are holding points, $S_{hp} \subseteq S_{stop}$
s_j	:	Stop number j, $s_j \in S_{stop}$, $j \in \mathbb{N}$
$t_{i,j}$	:	Measured trip departure time of trip i at stop s_j; $t_{i,j} \in \mathcal{M}$.
$pass_{in,i,j}$	:	Number of passengers entering the bus on trip i at stop s_j.
$pass_{out,i,j}$	:	Number of passengers leaving the bus on trip i at stop s_j.
z	:	Predetermined upper bound for the total planned trip runtime.
u	:	Upper bound of delay that is considered as in time.
f^{wait}	:	Factor between passenger perception of waiting time at a stop and real waiting time; $f^{wait} \geq 1$.

We use the following decision variables:

p_j : Planned departure time at stop s_j; $p_j \geq 0$, $p_0 = 0$. These times are part of set $\mathcal{P}$.
$a_{i,j}$: Computed time when bus is ready to leave in trip measurement i at stop s_j when planned departure time p_j is used; $a_{i,j} \in \mathcal{C}$.
$d_{i,j}$: Computed departure time in trip measurement i at stop s_j when planned departure time p_j is used; $d_{i,j} \in \mathcal{C}$.

Theoretically, we can determine planned departure times at any non-negative value in $\mathbb{R}$, but in information to passengers, only values rounded to minutes are used. Recall that we may choose to use $p_j \in \mathbb{R}$, $p_j \geq 0$ in our optimization models and round afterwards, or to restrict the domain and use $p_j \in \mathbb{N}$.

3.3.2 Constraints of the TRIP RUNTIME DETERMINATION PROBLEM

We define the constraints needed to describe the feasible region of the optimization problem using the variables from the previous section. First of all, we require that every passenger who enters the bus will eventually leave the bus, so for all trip measurements i, we have $\sum_j pass_{in,i,j} = \sum_j pass_{out,i,j}$.

When optimizing, our goal is to determine the times in set $\mathcal{P}$, so our main decision variables are the variables p_j. The variables $a_{i,j}$ for arrival times and $d_{i,j}$ for departure times in set $\mathcal{C}$ are calculated from measured departure times $t_{i,j}$ and planned departure times p_j as follows:

$$a_{i,j} = \begin{cases} 0 & \text{when } j = 0 \\ d_{i,j-1} + t_{i,j} - t_{i,j-1} & \text{otherwise} \end{cases} \tag{3.1}$$

$$d_{i,j} = \begin{cases} a_{i,j} & \text{when } s_j \notin S_{hp} \\ \max(a_{i,j}, p_j) & \text{when } s_j \in S_{hp} \end{cases} \tag{3.2}$$

Prolonging a trip costs money, because we use resources (vehicle, driver) for a longer time. When there is no cost component related to trip duration (for example when we only optimize on average delay or on number of tardy arrivals), a long planned trip duration will eventually lead to no delayed arrivals at all. Setting the stop passing time very high makes sure that all vehicles arrive on time. The fact that the vehicles have to wait very long is no component of the cost function, so this does not matter. Usually, we set the maximum on average total trip duration to a fixed value z, which is usually set to the 85th percentile value of the measured trip durations. This constraint on average trip duration can be formulated as follows:

$$z = \frac{\sum_i a_{i,n_{stop}-1}}{n_{trip}} \tag{3.3}$$

For this constraint and for several objectives in the next section we use the average value instead of the total value, because for the same problem or for the different runtime groups in the problem the number of measured trips may vary. In this way these formulas stay the same.

3.3.3 Objectives of the TRIP RUNTIME DETERMINATION PROBLEM

Some objectives that we will consider in our application of the TRIP RUNTIME DETERMINATION PROBLEM are:

- Maximize punctuality, which is defined as the percentage of delays of no more than u minutes for some $u > 0$,
- Minimize average delay of vehicles per stop, and
- Minimize average travel time per passenger.

The second objective seems to be the most logical one: we want to minimize delays. Because delays are usually inevitable, we want them to be as small as possible. As we will see, this criterion may result in a timetable where the punctuality at some stops is bad, while the punctuality at the rest of the stops is quite good. We also need to use an upper bound for the total trip runtime, because otherwise an average delay per stop of zero will be achievable by assigning a large runtime value between every two stops (cf. Section 3.3.2).

The most common method of measuring punctuality in the Netherlands corresponds to the first objective. We use the percentage of stop passages for which the delay does not exceed a certain amount of time. The actual delay is not taken into account, only the fact whether it is smaller or larger than a given threshold of u minutes. Also in this case we need to assign an upper bound to the total trip runtime, similar to the first objective.

Both objectives focus on delay per stop and do not take passengers into account. When part of a trip has only a few passengers, a delay will have less influence on the average travel time per passenger than a delay on a busy part of this trip. The third objective takes passengers into account, where the total travel time of a passenger is measured from the moment that he/she arrives at the bus stop (we assume that the passenger arrives at the planned departure time that is to be determined) until the arrival at his/her destination. Optionally, we can apply an additional factor for the time that the passenger has to wait at the departure stop for a delayed bus in order to represent the fact that waiting time is perceived longer than travel time.

The delay and travel times are influenced by the actual arrival and departure times on the stops. For the holding points, the departure time depends on the planned departure time, since vehicles that arrive too early have to wait for the planned departure times. For the other stops we can influence the actual times only indirectly by changing the planned departure times at the earlier holding points. In van Oort [2011], it is shown that changing the planned departure times at holding points can be used to reduce the variance of trip departure/arrival times.

3.3.4 Cost function of the TRIP RUNTIME DETERMINATION PROBLEM

In this section we will translate the objectives from Section 3.3.3 into a cost function. For determining the quality of a set of planned passing times, we use the following objectives:

- *Number of tardy arrivals*. Most current contracts for public transport use this measure for evaluation of punctuality, for example: the goal is to be no more than 5 minutes late in at least 85% of arrivals at stops. This is formulated as follows:

$$\frac{\sum_{i,j} \mathbf{1}_{(u,\infty)} \left[\max(0, a_{i,j} - p_j)\right]}{n_{trip} n_{stop}} \tag{3.4}$$

  where the part $\max(0, a_{i,j} - p_j)$ gives the arrival delay of trip i at stop j and where the function $\mathbf{1}_{(u,\infty)}[t]$ is the indicator function which gives 1 if $t \in (u, \infty)$ and 0 otherwise.

- *Average delay*. The disadvantage of the previous indicator is that it does not prefer small delays over longer delays as long as they are both less than the threshold value, or both longer than this value. Using the average delay will minimize these delays. The average delay is expressed as:

$$\frac{\sum_{i,j} \max(0, a_{i,j} - p_j)}{n_{trip} n_{stop}} \tag{3.5}$$

- *Average passenger travel time*. We define the average passenger travel time as the duration of an average travel of a passenger, which is measured from the planned departure time of the trip to the actual arrival time. The first part between planned departure time and actual departure time is the time that a passenger has to wait for a delayed bus. The part from the actual departure time to the actual arrival time is the actual travel time. The waiting time at a stop is multiplied by a factor f^{wait} because a passenger perceives waiting time longer than travel time.

  This average passenger travel time is split into two parts: the waiting time at departure and the actual travel time. The total waiting time at departure can be expressed as

$$\sum_{i,j} pass_{in,i,j} \max(0, d_{i,j} - p_j) \tag{3.6}$$

  The total travel time for all passengers is calculated by:

$$\sum_{i,j} (pass_{out,i,j} a_{i,j} - pass_{in,i,j} d_{i,j}) \tag{3.7}$$

  So the average passenger travel time can be expressed as

$$\frac{\sum_{i,j} (f^{wait} pass_{in,i,j} \max(0, d_{i,j} - p_j) + pass_{out,i,j} a_{i,j} - pass_{in,i,j} d_{i,j})}{\sum_{i,j} pass_{out,i,j}} \tag{3.8}$$

For the cost function we will use one of these objectives or a non-negative weighted sum of these values. We can also discard the division by n_{stop}, n_{trip} or $\sum_{i,j} pass_{out,i,j}$ in order to get the total duration or total delay.

3.4 Percentile method

The first method we use in the benchmark with our own methods is the *percentile method*. This method is currently used by most public transport companies in the Netherlands.

The runtimes at a stop are calculated by taking a percentile value of all measured driving times from the start of the trip. We use the values that are used in Section 2.1 of van Oort [2011]: the 85th percentile value for the total planned trip duration, the 50th percentile value for all holding points and the 35th percentile value for the rest of the stops. When we choose to use only values rounded to minutes as planned departure times, we round these calculated times to the nearest minute, values at exactly half minutes are rounded up.

When two subsequent stops have a different percentile value associated to them, and the second stop has a lower percentile value, it may occur that the planned departure time of the second stop is earlier than the planned departure time of the first stop. When this happens, we adjust the time at the second stop and set it to a value equal to the first stop.

Because the Percentile Method only uses a predetermined percentage and measured runtimes, which are not influenced by the planned departure times we are determining, it does not depend on a cost function.

In Section 3.8 we will compare this method with the other methods in this chapter.

3.5 Optimization using INTEGER LINEAR PROGRAMMING

The next method we discuss aims at determining the optimal set of planned departure times by defining the problem as an INTEGER LINEAR PROGRAMMING problem. With this formulation we can use an external solver to determine the optimal runtimes. The approach we follow is inspired by the method published in Kroon et al. [2007], where the authors set the total trip duration to a fixed value and determine the optimal planned departure times for every stop. We will add the handling of stops that are not holding points.

3.5.1 Definition of variables and parameters

In addition to the variables in section 3.3, we use the following notation:

M	:	An arbitrary big number, $M > z$.
f^{early}	:	A parameter that is equal to the relative weight of earliness relative to delay. One minute too early is weighted the same as f^{early} minutes too late.
$vc_{i,j}$	:	Binary variable that indicates whether the vehicle does wait or not for departure time at stop s_j of trip i: $vc_{i,j} \in \{0,1\}$ and $vc_{i,j} = 1$ indicates that the vehicle does not wait.
$vw_{i,j}$	:	Binary variable that indicates whether the vehicle waits for departure time at stop s_j of trip i: $vw_{i,j} \in \{0,1\}$ and $vw_{i,j} = 1$ indicates that the vehicle waits. We have that $vc_{i,j} + vw_{i,j} = 1$.
$l_{i,j}$	:	Binary variable that indicates whether the vehicle is late ($d_{i,j} > p_j + u$) at stop s_j of trip i: $l_{i,j} \in \{0,1\}$ and $l_{i,j} = 1$ if the vehicle is late.
$e^+_{i,j}$	:	Delay of vehicle at stop s_j of trip i: $e^+_{i,j} \in \mathbb{R}$, $e^+_{i,j} \geq 0$.
$e^-_{i,j}$	:	Earliness of vehicle at stop s_j of trip i: $e^-_{i,j} \in \mathbb{R}$, $e^-_{i,j} \geq 0$.

Although $vc_{i,j}$ and $vw_{i,j}$ are closely related, we have included both for ease of notation.

3.5.2 Constraints

For the INTEGER LINEAR PROGRAMMING formulation, we first define the constraints. The optimal values of the variables p_j, $a_{i,j}$, $d_{i,j}$ and $l_{i,j}$ are to be determined. We start with the constraints that are used to ensure that the actual departure times $d_{i,j}$ get the correct value. The actual departure time at a stop that is a holding point is never earlier than the planned passing time, and is never earlier than the actual departure time from the previous stop plus the measured driving time from the previous stop:

$$d_{i,j} \geq p_j, \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \in S_{hp} \tag{3.9}$$

$$d_{i,j} \geq d_{i,j-1} + (t_{i,j} - t_{i,j-1}), \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \in S_{hp} \tag{3.10}$$

$$d_{i,j} = d_{i,j-1} + (t_{i,j} - t_{i,j-1}), \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \notin S_{hp} \tag{3.11}$$

The planned departure time should be a non-negative integer:

$$p_j \geq 0, p_j \in \mathbb{N}, \forall j = 1, .., n_{stop} \tag{3.12}$$

The planned departure time on a stop should never be earlier than the planned departure time at the previous stop:

$$p_j \geq p_{j-1}, \forall j = 1, .., n_{stop} \tag{3.13}$$

At any holding point, a driver has to wait for the planned departure time, but he is not allowed to wait longer. Waiting longer at a holding point could prevent a measured trip to pass a future non-holding point too early. This implies that a driver knows exactly what will happen in the future, so this is not allowed. This is enforced using the following constraints:

$$d_{i,j} - Mvc_{i,j} \leq p_j, \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \in S_{hp} \tag{3.14}$$

$$d_{i,j} - Mvw_{i,j} \leq d_{i,j-1} + (t_{i,j} - t_{i,j-1}), \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \in S_{hp} \tag{3.15}$$

$$vw_{i,j} = 1 - vc_{i,j}, \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \in S_{hp} \tag{3.16}$$

Note that Equation (3.14) does nothing when $vc_{ij} = 1$ and changes Equation (3.9) into an equality when $vc_{i,j} = 0$. For Equations (3.15), (3.10) and variable $vw_{i,j}$, the opposite holds. The last three constraints are only necessary when there are stops $s \notin S_{hp}$ after stop $s_j \in S_{hp}$, since a possible benefit from waiting to leave after the planned departure time only occurs at non-holding points.

The total planned trip duration may be part of the cost function, or is determined before optimization. If the maximum trip runtime is set to z, we formulate this as:

$$p_{n_{stop}-1} - p_0 \leq z \tag{3.17}$$

For minimizing the number of delays, we need to define $l_{i,j}$.

$$d_{i,j} - Ml_{i,j} \leq p_j + u, \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \in S \tag{3.18}$$

For optimizing the minimum average delay, we need to define $e^+_{i,j}$ and $e^-_{i,j}$, which contains the deviation from the planned runtime:

$$d_{i,j} - e^+_{i,j} + e^-_{i,j} = p_j, \forall i = 0, .., n_{trip} - 1; \forall j \geq 1 : s_j \in S \tag{3.19}$$

3.5.3 Objective function

Finally we formulate the cost function. In Section 3.3.4, we formulated three different objectives. Each of them can be used to obtain a different INTEGER LINEAR PROGRAMMING model. We show here four different formulas, one for each objective in Section 3.3.4 and an additional one to represent the average trip duration. These formulas can be used in any linear combination with non-negative weights:

- *Number of tardy arrivals*. For this component, we use:

$$\text{Minimize} \quad \sum_{i,j} l_{i,j} \tag{3.20}$$

- *Average delay*. This cost component adds all delays. To get the real average delay, we divide it by $n_{trip} \times n_{stop}$. In order to reduce early stop passages, we also add $e^-_{i,j}$ with a factor f^{early}, which represents that one minute too early is f^{early} times worse than one minute too late. Equation 3.21 is not purely an average delay anymore, but a measure for deviation from the planned times.

$$\text{Minimize} \quad \frac{\sum_{i,j}(e^+_{i,j} + f^{early} e^-_{i,j})}{n_{trip} \times n_{stop}} \tag{3.21}$$

- *Average passenger travel time*. For passenger travel time, we calculate the total passenger travel time, where wait times at departure are multiplied by a factor f^{wait}. Dividing this figure by $\sum_{i,j} pass_{out,i,j}$ gives the average passenger travel time.

$$\text{Minimize} \quad \frac{\sum_{i,j} pass_{out,i,j} a_{i,j} - \sum_{i,j} pass_{in,i,j} d_{i,j} + f^{wait} \sum_{i,j} pass_{in,i,j}(d_{i,j} - p_j)}{\sum_{i,j} pass_{out,i,j}} \tag{3.22}$$

- *Average trip duration*. When the maximum total planned trip duration is not set to a fixed value in Equation (3.17), we can include it in the cost function as follows:

$$\text{Minimize} \quad p_{n_{stop}-1} \tag{3.23}$$

In Section 3.8 we will compare the use of INTEGER LINEAR PROGRAMMING with the other methods in this chapter.

3.5.4 A heuristic for an INTEGER LINEAR PROGRAMMING with holding points

An often occurring situation is that we want to determine optimal runtimes for a route where not all stops are holding points. When we relax the condition that p_j and $d_{i,j}$ are integral, $p_j \geq 0$ and $d_{i,j} \geq 0$, and also have that the objectives do not include the number of tardy arrivals, we see that in Equations (3.14), (3.15), and (3.16), $vc_{i,j}$ and $vw_{i,j}$ are the only variables in the INTEGER LINEAR PROGRAMMING formulation that are required to be integral. What is actually done in these equations, is to set the value of $d_{i,j}$ exactly equal to the maximum of p_j and $d_{i,j-1} + (t_{i,j} - t_{i,j-1})$. For our heuristic, we will get rid of these integer variables, in order to be able to solve the problem faster. We subsequently remove Equations (3.14), (3.15), and (3.16). Now Equations (3.9) and (3.10) can be expressed as $d_{i,j} \geq \max(p_j, d_{i,j-1} + (t_{i,j} - t_{i,j-1}))$. The drawback of this approach is that $d_{i,j}$ may get larger than $\max(p_j, d_{i,j-1} + (t_{i,j} - t_{i,j-1}))$, which occurs when it is better to wait longer than until the scheduled departure time at a holding point because it is beneficial further down the trip. For example, waiting may prevent earliness later and in this way improve passenger travel times. This is undesirable because in practice a driver does not know beforehand which disturbances he/she will encounter. In order to mitigate this effect, we add $d_{i,j}$ to the objective function. This will make the value of $d_{i,j}$ closer to the maximum of p_j and $d_{i,j-1} + (t_{i,j} - t_{i,j-1})$, but it is not guaranteed that it is equal to this maximum. Therefore the results of the INTEGER LINEAR PROGRAMMING heuristic are not guaranteed to be optimal. This is clearly to be seen in the results later in this chapter in Sections 3.8.3, 3.8.4 and 3.8.5, where we compare this method with the other methods in this chapter.

3.6 A heuristic based on DYNAMIC PROGRAMMING

Solving the TRIP RUNTIME DETERMINATION PROBLEM with integral planned departure times using the INTEGER LINEAR PROGRAMMING method as described in Sections 3.5.1 to 3.5.3 may be very slow, because of the large number of integral variables. In order to make real-world size problems tractable, we follow a different approach. We design a heuristic based on DYNAMIC PROGRAMMING, which we described earlier in Section 2.1. We use the basic variant, which is used for variables with discrete domains, like our variables p_j when they are integral.

For the DYNAMIC PROGRAMMING approach we use a model consisting of a grid with size $n_{stop} \times z$, where n_{stop} denotes the number of stops in a trip and z denotes the predetermined

upper bound for the total planned trip runtime. We use $c_{n,t}$ to denote the minimum cost belonging to trip stop n with $n \in \{0, n_{stop} - 1\}$ and planned departure time t measured from the start of the trip. $c^+_{n,t',t}$ denotes the extra cost when going from stop $n-1$ with planned departure time t' to stop n with planned departure time t. Furthermore, $d_{i,n,t}$ denotes the computed departure time of trip i at stop n with $n \in \{0, n_{stop} - 1\}$. When the DYNAMIC PROGRAMMING algorithm has finished with the calculation of stop $n-1$, all minimal costs and shortest paths of stop $n-1$ for all times are known. When calculating the extra cost $c^+_{n,t',t}$, we look at all trips at stop $n-1$ with planned departure time t' and calculate the new trip arrival and departure times for stop n with planned departure time t. We observe that the objective functions we defined in Section 3.5.3 are sums of costs on a stop depending on actual arrival times at that stop. For the calculation of $c^+_{n,t',t}$ we only have to process the part that is applicable to stop n.

For every pair of stop n and planned departure time t, we calculate $c_{n,t}$ once and $d_{i,n,t}$ for every measured trip and store these values. For this, we use the recursive relations from Equations (3.24) and (3.25), where the value t'_{min} in Equation (3.25) is the value for t' for which in Equation (3.24) the minimum is attained. The value of t'_{min} is also stored. After executing this process for all values n and t, we determine the optimal planned departure time at the last stop as the value t for which $c_{n_{stop}-1,t}$ is minimal. The optimal planned departure times at the other stops are determined by tracing back from the last stop to the first stop using the stored values for t'_{min}. This heuristic will be evaluated and compared to other methods in Section 3.8.

$$c_{n,t} = \begin{cases} 0 & \text{when } n = 0 \wedge t = 0 \\ \min_{t' \in [0,t]}(c_{n-1,t'} + c^+_{n,t',t}) & \text{when } n > 0, t \leq z \\ \text{undefined} & \text{otherwise} \end{cases} \tag{3.24}$$

$$d_{i,n,t} = \begin{cases} 0 & \text{when } n = 0 \wedge t = 0 \\ d_{i,n-1,t'_{min}} + t_{i,n} - t_{i,n-1} & \text{when } s_n \notin S_{hp} \\ \max(p_n, d_{i,n-1,t'_{min}} + t_{i,n} - t_{i,n-1}) & \text{otherwise} \end{cases} \tag{3.25}$$

As we will see later in the computational results, our DYNAMIC PROGRAMMING approach is a heuristic and does not always give the optimal result. This is because in this case DYNAMIC PROGRAMMING does not comply with the requirement required to get an optimal solution with DYNAMIC PROGRAMMING: *optimal substructure*. In this case, *optimal substructure* would imply that when a path in our grid from A to C is optimal, then the part of this path from A to B (with B anywhere on the optimal path from A to C) is also an optimal path. This is not true in our case, because trip departure times are not only determined from measured runtimes, but are also influenced by the planned departure time on holding points, the variable that we are optimizing. A planned departure time on a holding point on the optimal path from A to B may influence the optimal planned departure times after B, so a change of this value may make the path from A to B suboptimal, while making A to C better.

The idea however is that this method will still yield good results, and it is much faster than using an ILP as described in Section 3.5. In section 3.8, we will evaluate the DYNAMIC PROGRAMMING approach.

3.7 Continuous optimization

When looking back at the INTEGER LINEAR PROGRAMMING method, we see that the number of variables is $\mathcal{O}(n_{trips} \times n_{stops})$, and when not all stops are holding points, an additional $n_{trips} \times n_{hp}$ binary variables are added (see (3.14)-(3.16)). Furthermore, when the number of late departures is used in the objective function, there is an additional binary variable for every stop on every trip. This number can be quite large and experiments show that this leads to a long running time when optimizing. In this section, we do not require that the variables are integer and discuss a more efficient optimization method.

What we notice, is that the passing times $d_{i,j}$ besides by the measured trip departure times $t_{i,j}$ can only be influenced by the planned departure times p_j at holding points.

We will argue that for each stop that is not a holding point, we have that the optimal planned departure time is a function of all departure times from this stop. Because these departures are a function of the planned departures at holding points, this optimal planned departure time is also a function of the planned departures at holding points. This implies that the whole trip runtime determination problem can be expressed as an optimization problem with n_{hp} variables. In the remainder of this section, we will reformulate the TRIP RUNTIME DETERMINATION PROBLEM in terms of the variables p_j for the holding points.

Below we will show how the TRIP RUNTIME DETERMINATION PROBLEM can be solved using continuous optimization. However, this can only be done if the variables do not have to be integral. Therefore, this approach does not work when we want to minimize the number of tardy arrivals. We will describe our approach for objective functions depending on the delay; objectives with passenger travel time or total trip duration can be dealt with in a similar way.

To be able to work with different objectives, we use a general cost function $f(x)$ in our optimization algorithm, where x is the tardiness of a vehicle at a stop. This general cost function is described in Equation (3.26), where $f(x)$ is the cost function for one trip at one stop. In Section 3.7.2 we will define $f(x)$ in such a way that it reflects the average delay.

$$\text{Minimize} \sum_{i,j} f(d_{i,j} - p_j) \tag{3.26}$$

The TRIP RUNTIME DETERMINATION PROBLEM has $n_{trip} \times n_{stop} + n_{stop}$ variables, where we observed earlier that only the variables p_j with $s_j \in S_{hp}$ are relevant, and the values of the other variables follow from the values of these p_j variables. In the following paragraph we will reformulate the rest of the variables using only these variables p_j.

We start with the $d_{i,j}$-variables. For every stop on every trip, this departure time is equal to the departure time at the previous stop plus the measured driving time from the previous stop. When the current stop is a holding point and the departure time is earlier than the planned time, the planned time will be used for the departure time. The variable $d_{i,j}$ is defined recursively. In formula:

$$d_{i,j} = \begin{cases} 0 & \text{when } j = 0 \\ d_{i,j-1} + (t_{i,j} - t_{i,j-1}) & \text{when } s_j \notin S_{hp} \\ \max(d_{i,j-1} + (t_{i,j} - t_{i,j-1}), p_j) & \text{when } s_j \in S_{hp} \end{cases} \tag{3.27}$$

Now consider p_j where $s_j \notin S_{hp}$. We observe that the optimal value can be computed from the values $d_{i,j}$ for this j by determining p_j where $\sum_i f(d_{i,j} - p_j)$ is minimal and hence only depends on these values.

3.7.1 Formulation as convex optimization problem

As we have seen in Section 3.3, the variables p_j for the holding points are the ones that must be determined in the optimization problem. All the other variables are either parameters or can be calculated from measured trips departure times $t_{i,j}$ and these variables p_j.

In this section, we show that the objective function in Equation (3.26) is a convex function of variables p_j, unless the number of tardy arrivals is used in the cost function.

Theorem 1. *The actual arrival and departure times $a_{i,j}$ and $d_{i,j}$ of a trip are convex functions of the planned departure times p_j.*

Proof. Equations (3.1) and (3.2) form a set of recursive functions that define all arrival and departure times of a trip. Which case is used in each equation only depends on preset properties of stops and is independent of any variable that changes during the optimization, except for p_j. The arrival and departure times on stops are either a linear function of the previous departure time or a maximum of this linear function and a variable p_j. When we start at the second stop, the arrival time is a given number and the departure time is either this fixed number or the maximum of this number and a variable. In Section 3.2.3 of Boyd and Vandenberghe [2004], we see that the point-wise maximum of two convex function is convex. So when the second stop is a holding point, the departure time of the second stop is a convex function of the planned departure at this stop.

For the following stop, we know that the arrival time is the actual departure time of the previous stop plus the measured driving time. So the arrival time is a convex function of the planned departure times at preceding stops. When this stop is a holding point, then the departure time is the point-wise maximum of the arrival time (which is a convex function) and the planned departure time. We can continue this for all stops and conclude that convexity is preserved during the whole trip. So all actual arrival and departure times $a_{i,j}$ and $d_{i,j}$ are convex functions of p_j. □

Theorem 2. *The actual passenger waiting time $d_{i,j} - p_j$ for a vehicle of trip i at stop j is a convex function of the planned departure times p_j.*

Proof. For stops s_j that are not holding points, the waiting time is always zero, so for these the proof for convexity is trivial. When stop s_j is a holding point, the passenger waiting time is equal to $d_{i,j} - p_j$, which is non-negative as the vehicle is not allowed to depart before time p_j. Both the functions $d_{i,j}$ and $-p_j$ are convex functions of variables p, so the sum of these is also convex. □

As stated at the end of Section 3.3, the cost function is a linear combination of four possible objectives. For three of these objectives, we prove that the cost function is a convex function of these objectives. The number of tardy arrivals is not a convex function, so when we include this objective in the cost function, we can not guarantee that the cost function is still convex.

Theorem 3. *The cost function that is a non-negative weighted sum of Equations* (3.3)*,* (3.5) *and* (3.8) *is convex in the planned departure times* p_j*.*

Proof. A non-negative weighted sum of convex functions is convex, see Section 3.2.1 of Boyd and Vandenberghe [2004]. In order to prove convexity for the cost function, we prove that all its components are convex.

Equation (3.3) is a non-negative weighted sum of variables $a_{i,j}$, and each $a_{i,j}$ is calculated using a convex function from p_j. So Equation (3.3) is convex.

In Equation (3.5), the delay is calculated using a $\max$-function from a fixed value 0 and $a_{i,j} - p_j$. This last function is the sum of $a_{i,j}$ and $-p_j$, which are both convex, so $a_{i,j} - p_j$ is also convex. The $\max$-function of two convex function preserves convexity, so Equation (3.5) is a convex function of variables p_j.

Equation (3.8) consists of a non-negative weighted sum of Equations (3.6) and (3.7). The first part contains $\max(0, d_{i,j} - p_j)$, which is convex using a similar argument as used with Equation (3.5). The weighted sum with positive weights of all these elements preserves convexity. The convexity of Equation (3.7) can be proved by splitting the travels in two parts: the actual travel times between departure from a stop to the arrival at the next stop and the actual waiting time at stops. The actual travel time is a fixed number, only depending on actual travel times between stops and not dependent on any planned departure time p_j. The actual waiting time is a convex function, as proved in Theorem 2. So the duration of all waiting times is convex and the sum of all passenger travels is also convex. □

This all leads to the next theorem:

Theorem 4. *When the* TRIP RUNTIME DETERMINATION PROBLEM *with a cost function that is a non-negative weighted sum of Equations* (3.3)*,* (3.5) *and* (3.8) *has a local minimum, this local minimum is also a global minimum.*

Proof. In Theorem 3, we showed that the cost function of the TRIP RUNTIME DETERMINATION PROBLEM is convex in p_j for all cost components except the number of tardy passages. For a convex function, we know that when there exists a local minimum, this also a global minimum. □

Next we generalize to the following theorem:

Theorem 5. *Consider the* TRIP RUNTIME DETERMINATION PROBLEM *with objective function* (3.26)*. When* $f(x)$ *is convex and non-decreasing, the objective function* (3.26) *is convex and consequently this* TRIP RUNTIME DETERMINATION PROBLEM *is a convex optimization problem.*

Proof. Recall that we work with the general objective function in Equation (3.26). This equation can be considered as a sum of composite functions $f(g(p))$, where $g(p) = d_{i,j} - p_j$ with $d_{i,j}$ as defined in Equation (3.27) is a convex, non-decreasing function. This composition of functions results in a convex function if and only if $f(x)$ is convex and non-decreasing and $g(p) = d_{i,j} - p_j$ is convex, see Section 3.2.4 of Boyd and Vandenberghe [2004]. We have already shown that $d_{i,j} - p_j$ is convex, so when $f(x)$ is convex and non-decreasing, the objective function (3.26) is convex. □

Convexity is a desirable property for us, because there exist efficient methods to optimize a convex function, and we know that the optimization will end in a global minimum because of the absence of other local minima.

3.7.2 Algorithm for minimizing total weighted earliness and delay

For our experimental study, we want to focus on delay. However, besides leaving too late, leaving too early is also undesirable. Therefore we specify a cost function $f(x)$ including both aspects. This has a drawback that $f(x)$ is only almost convex. We will explain how to deal with that.

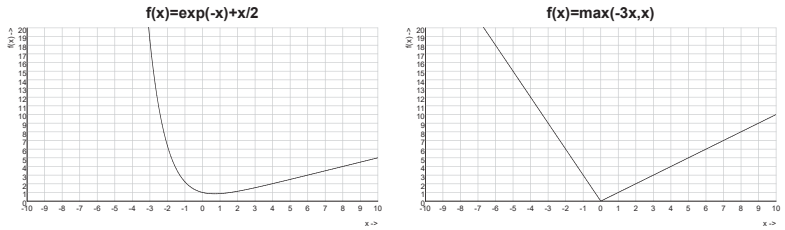
The function $f(x)$ should comply to the following:

- Because delay is considered as undesired, $f(x)$ should be increasing for $x > 0$
- Because leaving too early is very undesirable, $f(x)$ should be decreasing when $x < 0$, the cost of being too early has to be higher than the cost of being too late.
- A continuous function is preferable, in order to make the optimization tractable. A differentiable function is even more preferable.

We have chosen two different expressions for $f(x)$, which are shown in Figures 3.5a and 3.5b:

$$f(x) = e^{-x} + \frac{x}{2} \tag{3.28}$$

$$f(x) = \max(-3x, x) = \begin{cases} x & \text{when } x \geq 0 \\ -3x & \text{when } x < 0 \end{cases} \tag{3.29}$$



(a) Graph of Equation (3.28) (b) Graph of Equation (3.29)

Figure 3.5: Graph of Equations (3.28) and (3.29)

In both expressions, we penalize for being too late and for being too early. Being too late is penalized with a linear cost, being too early is generally considered as undesirable and is penalized much more than being too late. In Equation (3.28), we use a combination of an exponential function with a linear function. This does comply to our wishes and the function is continuously differentiable. Because both functions are not non-decreasing, we

can not guarantee that Equation (3.26) is convex and that we can use *convex optimization*. Because the most important part of both functions, the part where the vehicle is too early, is non-increasing, we try this approach anyway and perform calculations in order to test the usability of this solution method of the TRIP RUNTIME DETERMINATION PROBLEM.

We also note that this optimization does not have constraints: it is unbounded optimization. Furthermore, the objective function is continuous. We choose to implement the optimization method as proposed by Powell [1964], a method that uses an iterative process to find the optimal value handling continuous and convex or almost-convex objectives like ours very well. It does not require the objective to be differentiable, so it is also applicable when we use Equation (3.29) as definition for $f(x)$. We will evaluate the use of continuous optimization using both definitions for $f(x)$ in Section 3.8.

3.8 Computational results

For the evaluation of all earlier mentioned methods, we use four different datasets, containing real runtime measurements and passenger counts for four different kinds of routes in the north of the Netherlands:

- *Route 6*. This is an urban route in the city of Groningen. It runs from one side of the city to the other side and runs through the busy center of the city. See Figure 3.6.
- *Route 39*. Route 39 is a suburban route from Groningen to a lot of small villages, ending in Surhuisterveen. See Figure 3.7.
- *Route 50*. This suburban route runs between two large cities (Assen and Groningen) and services some larger villages in between. See Figure 3.8a.
- *Route 300*. This is an express-bus from Groningen to Emmen, which has only a few stops and mainly drives on highways. See Figure 3.8b.

For every route we have a dataset for every direction. In Table 3.1, we put the characteristics of every dataset. We use data for weekdays, so for all datasets we split the day into parts for which we separately determine the optimal runtimes. These parts are the runtime groups. In this column we denote the runtime group boundaries, that is, the times when the next runtime group starts. In the next columns, we denote the number of measured trips, the number of stops of every trip and the number of holding points during the trip.

For this comparison, we use data for the earlier mentioned routes from 13 May 2013 to 28 June 2013. We only used the trips where for every stop at least the departure was registered. We also removed some trips which contained evident measurement errors caused by malfunctioning equipment.

For the passenger counts we used the data from the electronic ticketing system of the vehicles, which registered every time a passenger leaves or enters the vehicle. In this way, we know for every trip between 13 May 2013 and 28 June 2013 how many passengers boarded and how many alighted at on every stop. We removed the measurements when a passenger

Figure 3.6: The purple line indicates route 6

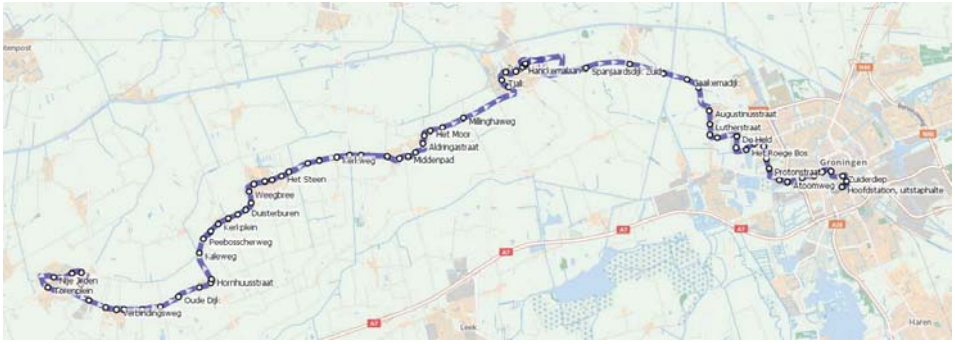
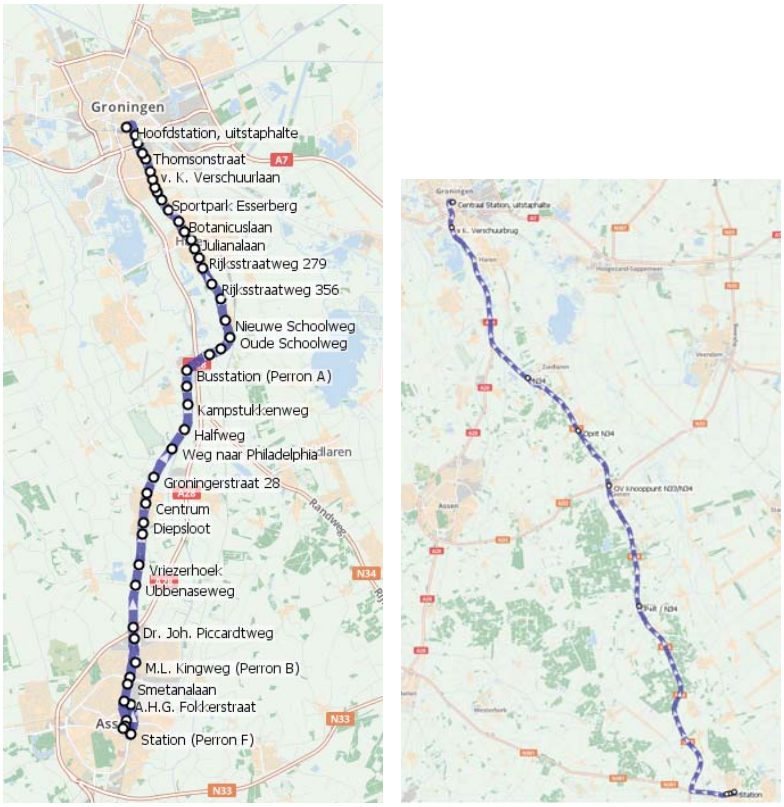


Figure 3.7: The purple line indicates route 39

did not both enter and leave the vehicle at some time. Furthermore, we remark that not all passengers use an electronic ticket. Our measurements are thus lower than the real number of passengers, but they give a good representation of the number of passengers.

The evaluation is split into two parts. The first part compares the optimizations on punctuality and average delay, using integral planned departure times. The second part compares the optimizations on average delay and average travel time where the planned departure times do not have to be integral. We have made this division because the DYNAMIC



(a) The purple line indicates route 50 (b) The purple line indicates route 300

Figure 3.8: Map of routes 50 and 300

PROGRAMMING method and the INTEGER LINEAR PROGRAMMING method can not work with non-integral times and the continuous optimization method only works with non-integral times. Non-integral times may be rounded to integral times, but this will inevitably lead to a worse result. For the continuous optimization method, we compared the non-integral and rounded non-integral solutions. This can be found at the end of Section 3.8.3.

Note that we did not test every possible combination of method and objective. We have only tested combinations that are necessary for our conclusions.

In Tables 3.2 to 3.16 we report the results of our experiments. The following data is put into the following tables:

- *Dataset*. The data used in the optimization
- *Route*. Description of the route
- *Method*. Method used for runtime determination

Dataset	Route	Runtime groups	#trips	#stop	#HP
6a	Bey-How	07:30/18:00	837	38	4
6b	How-Bey	07:30/18:00	810	38	5
39a	Shv-Gn	06:00/17:00	302	75	17
39b	Gn-Shv	14:45/18:00	225	72	15
50a	Asn-Gn	07:00/17:00	576	43	9
50b	Gn-Asn	07:00/19:00	538	42	9
300a	Emn-Gn	07:00/08:00/19:00	1241	9	7
300b	Gn-Emn	19:00	1235	9	7

Table 3.1: Properties of datasets used for runtime determination

- *Calc.dur.* Duration of the calculation of the optimization in seconds
- *Avg.delay.* Average delay, calculated over all stop departures in seconds.
- *Punct.* Percentage of trip departures that are not too early and between 0 and 3 minutes late, measured on all stops.
- *Punct.end.* Percentage of trip arrivals less than three minutes late, measured only on end stops.
- *Psg.trv.* Average passenger travel time inside the vehicle
- *Psg.wait.* Average passenger waiting time at the departure stop
- *Psg.dur.* Average perceived passenger travel time, this is a sum of the travel time inside the vehicle and twice the waiting time at the departure stop.

All optimizations use one single objective, except in Section 3.8.5, where a combination of two objectives is used. All optimizations are run on a computer with an Intel®Core™i7-3770 microprocessor running on 3.4 GHz, with 16 GB RAM memory. We implemented the algorithms in Java 8, using IBM ILOG CPLEX 12.2 for solving LPs and ILPs.

3.8.1 Optimization on punctuality with integral times

For the comparison on punctuality using integral times, we use $u = 3$ minutes. We compare the results for all eight datasets for the following:

- The current timetable, prior to optimization, as it is currently driven. Results can be found in Table 3.2.
- Optimization using the percentile-method as described in Section 3.4, rounding the results to the nearest integer. Results can be found in Table 3.3.
- Optimization using DYNAMIC PROGRAMMING as described in Section 3.6. Results can be found in Table 3.4.

We did not include the INTEGER LINEAR PROGRAMMING model from Section 3.5 because this method did not finish within twelve hours or before the computer ran out of memory.

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Current	0.02	85.96	58.00	99.40	10.81	1.38	13.57
6b	How-Bey	Current	0.04	121.14	57.26	98.77	11.34	1.43	14.20
39a	Shv-Gn	Current	0.02	89.63	71.58	80.13	22.65	1.42	25.49
39b	Gn-Shv	Current	0.02	85.98	72.55	76.00	23.43	0.87	25.17
50a	Asn-Gn	Current	0.03	81.33	53.10	97.57	11.27	0.93	13.13
50b	Gn-Asn	Current	0.01	71.86	51.49	98.70	12.09	0.59	13.27
300a	Emn-Gn	Current	0.03	7.29	99.17	98.39	32.01	0.09	32.19
300b	Gn-Emn	Current	0.01	9.03	99.24	99.27	28.03	0.09	28.21

Table 3.2: Computational results for current timetable

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Perc-Int	0.02	103.25	75.97	98.45	9.45	1.38	12.21
6b	How-Bey	Perc-Int	0.02	103.13	75.51	97.16	9.55	1.47	12.49
39a	Shv-Gn	Perc-Int	0.02	122.00	75.43	98.68	22.80	1.92	26.64
39b	Gn-Shv	Perc-Int	0.02	128.00	73.44	96.44	22.59	1.40	25.39
50a	Asn-Gn	Perc-Int	0.02	84.58	94.09	97.57	10.67	1.24	13.15
50b	Gn-Asn	Perc-Int	0.02	86.78	82.92	98.51	11.14	0.91	12.96
300a	Emn-Gn	Perc-Int	0.02	52.53	95.70	97.50	29.55	0.57	30.69
300b	Gn-Emn	Perc-Int	0.01	52.38	94.45	97.98	25.81	0.47	26.75

Table 3.3: Computational results for optimization using the percentile method and integral runtimes

When we compare the results for punctuality, we see that except for the datasets 300a and 300b, the Percentile-method performs better than the current timetable and that the DYNAMIC PROGRAMMING-method performs better than the Percentile-method. For datasets 300a and 300b, we see that the punctuality is a little worse than the current timetable, but the average travel time is minutes shorter.

When punctuality is the only objective, our experiments indicate that DYNAMIC PROGRAMMING should be used.

3.8.2 Optimization on average delay with integral times

For the comparison on average delay using integral times, we compare the results for all eight datasets for the following:

- The current timetable. Results can be found in Table 3.2.
- Optimization using the percentile-method. We use the results from Table 3.3.
- Optimization using DYNAMIC PROGRAMMING as described in Section 3.6. Results can be found in Table 3.5.

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	DP-Punct	2.62	104.39	80.03	98.09	9.48	1.49	12.46
6b	How-Bey	DP-Punct	2.56	91.18	87.59	96.30	9.90	1.29	12.48
39a	Shv-Gn	DP-Punct	5.23	111.37	75.96	98.68	22.68	1.97	26.62
39b	Gn-Shv	DP-Punct	2.01	116.81	76.35	96.89	22.88	1.30	25.48
50a	Asn-Gn	DP-Punct	2.45	83.99	89.42	96.35	10.58	1.24	13.06
50b	Gn-Asn	DP-Punct	2.46	83.14	89.72	97.77	11.35	0.85	13.05
300a	Emn-Gn	DP-Punct	0.94	37.46	97.44	97.90	29.68	0.35	30.38
300b	Gn-Emn	DP-Punct	1.26	62.41	92.84	98.14	25.79	0.55	26.89

Table 3.4: Computational results for optimization using the DYNAMIC PROGRAMMING method and integral runtimes

- Optimization using continuous optimization using the linear objective function from Equation (3.29). After the optimization, we round down all times. Results can be found in Table 3.6.

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	DP-Dly	2.62	84.11	69.39	98.09	9.48	1.14	11.76
6b	How-Bey	DP-Dly	2.77	66.49	73.72	96.30	9.90	0.93	11.76
39a	Shv-Gn	DP-Dly	5.80	120.86	61.93	98.01	22.69	2.15	26.99
39b	Gn-Shv	DP-Dly	1.93	96.86	67.45	96.89	22.88	1.09	25.06
50a	Asn-Gn	DP-Dly	2.24	65.45	76.03	96.35	10.58	0.98	12.54
50b	Gn-Asn	DP-Dly	2.66	62.70	77.13	97.77	11.35	0.62	12.59
300a	Emn-Gn	DP-Dly	0.81	38.31	96.73	97.74	29.65	0.39	30.43
300b	Gn-Emn	DP-Dly	1.21	62.41	92.84	98.14	25.79	0.55	26.89

Table 3.5: Computational results for optimization on average delay using the DYNAMIC PROGRAMMING method and integral runtimes

What we see, is that the use of DYNAMIC PROGRAMMING gives a better result than the percentile method, but the average delay when using continuous optimization with a linear objective function gives by far the best result. The only downside of this last method is the punctuality of the trip, which is much worse than the other methods. This is caused by the calculation of the average delay as we see in Equation (3.21): when a bus is a few seconds early, it will only have a small cost. When we measure punctuality, the trip is not considered on time anymore. Despite that the bus is only a few seconds early, the full cost for being too early will be applied.

3.8.3 Optimization on average delay with non-integral times

For the comparison on average delay using non-integral times, we compare the results for all eight datasets for the following:

- The current timetable. Although this one has integral planned times, we use them for comparison. Results can be found in Table 3.2.

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Clin-Dly-I	0.83	61.00	73.78	92.35	10.52	0.97	12.46
6b	How-Bey	Clin-Dly-I	1.04	60.27	79.12	80.99	10.83	0.68	12.19
39a	Shv-Gn	Clin-Dly-I	9.63	43.90	80.82	0.00	25.14	0.65	26.44
39b	Gn-Shv	Clin-Dly-I	4.38	41.74	73.48	3.11	25.50	0.57	26.64
50a	Asn-Gn	Clin-Dly-I	2.36	39.77	71.65	81.25	10.64	0.45	11.54
50b	Gn-Asn	Clin-Dly-I	2.47	40.39	70.72	87.17	11.84	0.37	12.58
300a	Emn-Gn	Clin-Dly-I	0.08	18.54	98.80	96.94	29.91	0.17	30.25
300b	Gn-Emn	Clin-Dly-I	0.08	26.19	97.69	97.57	26.14	0.27	26.68

Table 3.6: Computational results for optimization on average delay using continuous optimization with a linear objective function and runtimes rounded to integers

- Optimization using the percentile method. Results can be found in Table 3.7.
- Optimization using the INTEGER LINEAR PROGRAMMING heuristic, described in Section 3.5.4. Results can be found in Table 3.8.
- Optimization using continuous optimization using the nonlinear objective function from Equation (3.28). Results can be found in Table 3.9.
- Optimization using continuous optimization using the linear objective function from Equation (3.29). Results can be found in Table 3.10.

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Percent	0.02	89.62	76.17	97.25	9.51	1.13	11.77
6b	How-Bey	Percent	0.02	91.52	73.24	95.68	9.58	1.24	12.06
39a	Shv-Gn	Percent	0.02	107.95	80.46	95.36	22.69	1.75	26.19
39b	Gn-Shv	Percent	0.02	112.03	77.97	96.00	22.65	1.13	24.91
50a	Asn-Gn	Percent	0.02	70.00	89.05	96.18	10.48	1.01	12.50
50b	Gn-Asn	Percent	0.02	71.56	88.82	96.65	11.16	0.75	12.66
300a	Emn-Gn	Percent	0.04	33.26	97.70	97.66	29.40	0.33	30.06
300b	Gn-Emn	Percent	0.02	41.32	95.56	97.73	25.82	0.37	26.56

Table 3.7: Computational results for optimization using the percentile method and non-integral runtimes

In terms of average delay, the continuous optimization using a linear objective function gives the best results, but at the cost of punctuality at the end, especially on route 39, where almost no trip ends in time. The INTEGER LINEAR PROGRAMMING heuristic gives good results with good punctuality. In general this heuristic gives good results, and for some routes continuous optimization with a linear objective function gives better results with an acceptable end punctuality. For the other methods, we see results with much higher average delay than these two methods, so we prefer the continuous optimization, with a linear objective.

When we compare Table 3.6 with Table 3.10, we can compare the influence of rounding on the solution of optimization. We see that the average delay for the integral solution in

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	LP-Dly	2.44	52.40	77.76	69.30	10.71	0.70	12.11
6b	How-Bey	LP-Dly	1.74	51.86	77.75	49.63	10.95	0.55	12.05
39a	Shv-Gn	LP-Dly	1.70	33.10	79.98	1.66	24.17	0.49	25.15
39b	Gn-Shv	LP-Dly	1.22	30.76	80.83	0.00	25.46	0.42	26.30
50a	Asn-Gn	LP-Dly	2.02	28.17	81.06	47.74	10.88	0.28	11.44
50b	Gn-Asn	LP-Dly	1.48	27.85	80.60	31.04	12.49	0.25	12.99
300a	Emn-Gn	LP-Dly	0.58	13.19	98.93	96.54	30.01	0.10	30.21
300b	Gn-Emn	LP-Dly	0.52	19.00	98.07	96.76	26.30	0.15	26.60

Table 3.8: Computational results for optimization on average delay using the INTEGER LINEAR PROGRAMMING heuristic and non-integral runtimes

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Cnl-Dly	0.31	59.22	91.45	88.29	10.56	0.88	12.32
6b	How-Bey	Cnl-Dly	0.46	58.56	90.85	86.05	10.77	0.71	12.19
39a	Shv-Gn	Cnl-Dly	1.99	47.42	96.02	15.23	23.94	0.74	25.42
39b	Gn-Shv	Cnl-Dly	0.82	45.14	96.90	72.00	24.86	0.60	26.06
50a	Asn-Gn	Cnl-Dly	0.95	42.85	97.27	80.38	10.55	0.51	11.57
50b	Gn-Asn	Cnl-Dly	0.91	42.73	97.45	87.55	11.97	0.40	12.77
300a	Emn-Gn	Cnl-Dly	0.16	36.06	98.59	95.41	30.40	0.32	31.04
300b	Gn-Emn	Cnl-Dly	0.15	35.60	98.03	96.84	26.50	0.21	26.92

Table 3.9: Computational results for optimization on average delay using continuous non-linear optimization and non-integral runtimes

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Clin-Dly	0.76	48.89	76.89	80.17	10.61	0.70	12.01
6b	How-Bey	Clin-Dly	1.00	47.96	76.95	66.30	10.86	0.56	11.98
39a	Shv-Gn	Clin-Dly	9.82	27.33	81.15	0.00	25.21	0.40	26.01
39b	Gn-Shv	Clin-Dly	4.31	26.23	80.61	0.89	25.69	0.40	26.49
50a	Asn-Gn	Clin-Dly	2.40	27.25	81.10	76.04	10.67	0.30	11.27
50b	Gn-Asn	Clin-Dly	2.47	26.68	81.32	80.30	12.09	0.25	12.59
300a	Emn-Gn	Clin-Dly	0.05	13.19	98.94	96.54	30.01	0.10	30.21
300b	Gn-Emn	Clin-Dly	0.06	19.00	98.08	96.76	26.30	0.15	26.60

Table 3.10: Computational results for optimization on average delay using continuous optimization with a linear objective function and non-integral runtimes

Table 3.6 is only about 12 seconds more than the non-integral solution. The other figures are quite similar.

3.8.4 Optimization on average travel duration with non-integral times

For the comparison on average travel duration using non-integral times, we use the objective function from Equation (3.22). We compare the results for all eight datasets for the following:

- The current timetable. Although this one has integral planned times, we use them for comparison. Results can be found in Table 3.2.
- Optimization using the percentile method. Results are the same as with optimization on average delay because this method does not use an objective. The results can be found in Table 3.7.
- Optimization using the INTEGER LINEAR PROGRAMMING heuristic from Section 3.5.4. Results can be found in Table 3.11.
- Optimization using continuous optimization using a nonlinear objective function. Results can be found in Table 3.12.
- Optimization using continuous optimization using a linear objective function. Results can be found in Table 3.13.

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	LP-Pass	3.37	63.04	63.20	97.73	9.85	0.95	11.75
6b	How-Bey	LP-Pass	2.48	58.55	66.74	95.68	10.08	0.75	11.58
39a	Shv-Gn	LP-Pass	2.53	38.97	70.33	60.26	23.07	0.69	24.45
39b	Gn-Shv	LP-Pass	1.71	44.02	73.21	86.22	23.51	0.62	24.75
50a	Asn-Gn	LP-Pass	2.30	30.56	71.15	84.90	10.40	0.37	11.14
50b	Gn-Asn	LP-Pass	1.81	34.04	74.16	94.98	11.41	0.33	12.07
300a	Emn-Gn	LP-Pass	0.77	24.00	98.11	97.02	29.67	0.15	29.97
300b	Gn-Emn	LP-Pass	0.85	36.92	96.57	97.89	25.96	0.20	26.36

Table 3.11: Computational results for optimization on average travel duration using the INTEGER LINEAR PROGRAMMING heuristic and non-integral runtimes

Here we see that the continuous optimization with a linear objective function gives the best results, but at the cost of average delay and punctuality. When we compare the overall results from Tables 3.10 and 3.13, we see that optimization on average delay gives a passenger travel time that is only a few percent higher than when optimizing on passenger travel time.

3.8.5 Optimization on average delay and travel duration with non-integral times

In the last two sections, we used the average delay and the average travel duration as the objectives to optimize. In this section, we will optimize on the weighted sum of these two

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Cnl-Pass	3.42	153.55	72.65	98.21	9.45	1.18	11.81
6b	How-Bey	Cnl-Pass	2.84	87.07	77.19	96.91	9.67	1.11	11.89
39a	Shv-Gn	Cnl-Pass	29.27	138.86	80.57	94.37	22.56	1.23	25.02
39b	Gn-Shv	Cnl-Pass	11.11	819.08	46.23	96.44	22.40	1.14	24.68
50a	Asn-Gn	Cnl-Pass	8.08	69.37	88.31	96.88	10.53	0.88	12.29
50b	Gn-Asn	Cnl-Pass	5.77	126.08	76.65	98.33	11.09	0.70	12.49
300a	Emn-Gn	Cnl-Pass	0.14	76.29	89.38	98.39	29.53	0.63	30.79
300b	Gn-Emn	Cnl-Pass	0.18	94.26	84.94	98.22	25.73	0.56	26.85

Table 3.12: Computational results optimization on average travel duration using continuous non-linear optimization and non-integral runtimes

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Clin-Pass	1.31	129.28	68.55	96.77	9.76	0.83	11.42
6b	How-Bey	Clin-Pass	1.62	61.14	72.14	87.41	10.20	0.66	11.52
39a	Shv-Gn	Clin-Pass	13.21	111.77	74.98	77.15	22.82	0.71	24.24
39b	Gn-Shv	Clin-Pass	1.82	809.55	50.18	94.22	22.83	0.69	24.21
50a	Asn-Gn	Clin-Pass	3.26	43.19	78.81	93.06	10.37	0.42	11.21
50b	Gn-Asn	Clin-Pass	2.43	96.14	74.53	96.10	11.26	0.38	12.02
300a	Emn-Gn	Clin-Pass	0.41	38.15	96.46	97.82	29.61	0.25	30.11
300b	Gn-Emn	Clin-Pass	0.06	49.70	95.21	97.98	25.90	0.24	26.38

Table 3.13: Computational results optimization on average travel duration using continuous optimization with a linear objective function and non-integral runtimes

objectives, where the average delay is weighted double in relation to the travel duration. The idea is that we end up with a solution that does well on both average delay and travel duration.

For the comparison we add the objective functions from Equations (3.21) and (3.22), where Equation (3.21) is first doubled. We compare the results for all eight datasets for the following:

- The current timetable. Although this one has integral planned times, we use them for comparison. Results can be found in Table 3.2.
- Optimization using the percentile method. Results are the same as with optimization on average delay because this method does not use an objective. The results can be found in Table 3.7.
- Optimization using the INTEGER LINEAR PROGRAMMING heuristic from Section 3.5.4. Results can be found in Table 3.14.
- Optimization using continuous optimization using the nonlinear objective function from Equation (3.28). Results can be found in Table 3.15.
- Optimization using continuous optimization using the piecewise linear objective function from Equation (3.29). Results can be found in Table 3.16.

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	LP-D+P	5.16	52.23	71.22	95.70	10.18	0.78	11.74
6b	How-Bey	LP-D+P	2.63	50.74	73.33	94.32	10.40	0.62	11.64
39a	Shv-Gn	LP-D+P	4.52	34.86	73.21	47.02	23.35	0.59	24.53
39b	Gn-Shv	LP-D+P	2.18	35.84	75.29	82.22	23.99	0.57	25.13
50a	Asn-Gn	LP-D+P	2.94	28.08	77.09	81.60	10.48	0.33	11.14
50b	Gn-Asn	LP-D+P	2.63	28.65	77.98	91.26	11.64	0.28	12.20
300a	Emn-Gn	LP-D+P	1.02	16.17	98.58	96.70	29.78	0.11	30.00
300b	Gn-Emn	LP-D+P	1.13	22.07	97.79	97.09	26.10	0.16	26.42

Table 3.14: Computational results optimization on average delay and travel duration using the linear programming heuristic and non-integral runtimes

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Cnl-D+P	8.96	70.05	84.92	97.73	9.69	1.03	11.75
6b	How-Bey	Cnl-D+P	8.20	68.49	85.40	96.42	9.91	0.92	11.75
39a	Shv-Gn	Cnl-D+P	72.74	61.98	89.76	92.72	22.62	1.03	24.68
39b	Gn-Shv	Cnl-D+P	15.56	74.98	80.23	93.78	22.79	0.87	24.53
50a	Asn-Gn	Cnl-D+P	19.93	52.06	93.34	95.66	10.44	0.71	11.86
50b	Gn-Asn	Cnl-D+P	14.30	56.46	91.25	97.77	11.19	0.55	12.29
300a	Emn-Gn	Cnl-D+P	0.19	54.01	95.48	98.23	29.56	0.49	30.54
300b	Gn-Emn	Cnl-D+P	0.22	54.73	94.40	98.14	25.80	0.39	26.58

Table 3.15: Computational results optimization on average delay and travel duration using continuous optimization using the non-linear objective function from Equation (3.28) and non-integral runtimes

Dataset	Route	Method	Calc.dur.	Avg.delay	Punct.	Punct.end	Psg.trv	Psg.wait	Psg.dur.
6a	Bey-How	Clin-D+P	2.57	51.79	75.70	95.82	10.08	0.75	11.58
6b	How-Bey	Clin-D+P	3.09	50.22	75.93	92.22	10.45	0.60	11.65
39a	Shv-Gn	Clin-D+P	14.35	34.77	79.39	58.94	23.20	0.58	24.36
39b	Gn-Shv	Clin-D+P	5.28	39.61	78.54	84.89	23.60	0.53	24.66
50a	Asn-Gn	Clin-D+P	5.30	28.82	80.73	87.50	10.39	0.36	11.11
50b	Gn-Asn	Clin-D+P	5.78	30.53	80.72	93.49	11.51	0.30	12.11
300a	Emn-Gn	Clin-D+P	0.09	19.01	98.57	97.34	29.76	0.14	30.04
300b	Gn-Emn	Clin-D+P	0.10	21.50	97.91	97.09	26.14	0.15	26.44

Table 3.16: Computational results optimization on average delay and travel duration using continuous optimization with the piecewise linear objective function from Equation (3.29) and non-integral runtimes

When we compare the results, we see that the results from the percentile-method are worse than the other methods. For the INTEGER LINEAR PROGRAMMING heuristic and the two continuous optimization models, the average passenger travel durations are comparable to each other. However, the average delay for continuous optimization using the non-linear objective function results in a higher average delay than for the other two methods. These other two methods, the INTEGER LINEAR PROGRAMMING heuristic and continuous optimization using the linear objective function, both give the best results for average delay and travel duration, but continuous optimization using the linear objective function results in a punctuality that is a few percent higher. So we conclude that continuous optimization with a linear objective function gives the best result.

3.9 Runtime groups

3.9.1 Introduction

In the previous sections, we have developed several methods to solve the TRIP RUNTIME DETERMINATION PROBLEM, starting with a given set of trips and trip measurements. In practice, we see that the measured runtimes change during the day. For example: when we look at Figure 3.1, we see that the driving times during the day are clearly higher than during the evening. In a rush hour, a trip will generally take more time than in the evening. To take this into account, we can divide the day in periods and determine the runtime for each of these periods. We call this a runtime group. The idea of using runtime groups is widely used and is discussed in van Oort [2011]. However, it is only used together with the percentile method. In this section, we will determine runtime groups using continuous optimization with a linear objective function with non-integral runtimes from Section 3.7, where we use the average perceived travel duration per passenger as objective.

3.9.2 Problem description

For the use of runtime groups, we have two extreme situations. On one side there is one runtime group for the whole day: the timetable is the same for the whole day. On the other side, we would determine a different timetable for every single trip: every runtime group contains one trip. Using only one runtime group per day has the advantage that the timetable is regular; at every hour of the day the departure times are the same. Unfortunately, passenger travel time will increase a lot, because in rush hours the passengers have to wait more for the buses because they will be often too late, whereas in the evening the buses will have to wait a lot at stops because they arrive too early. One runtime group per trip is not practical, because it leads to irregular passing times in the timetable, which makes it difficult to remember. We want to end up somewhere between these two extremes: n runtime groups, where the value for n is to be determined. When $n > 1$, we have to determine the optimal division in runtime groups. Where does each runtime group start and end?

3.9.3 Approach

For our experiments, we determine the optimal runtime groups for a number of different values of n. We start with $n = 1$, and increase n until $n = 8$, where we use whole hours as possible start and end times of the runtime groups. We determine optimal runtimes for every possible runtime group. The cost of a runtime group is determined as the total travel time for all passengers in that runtime group. Then we determine the optimal set of runtime groups that cover the day. We do this by building a directed, acyclic graph. For every combination of boundary of runtime groups and number of runtime groups, we create a node. We create arcs for every runtime group from a node representing the start of the runtime group to a node representing the end of the runtime group. In this way every path in the graph represents a sequence of runtime groups covering the day. The weight of an arc is the cost of this runtime group. By determining the shortest path in this graph, we can decide on the best set of runtime groups. The total cost is then divided by the number of passengers, in order to get the average passenger travel time.

3.9.4 Results

To test the influence of the number of runtime groups on passenger travel time, we determined the optimal runtime groups for all our datasets for weekdays with a predetermined runtime group count. The results are presented in Table 3.17. Furthermore, we plotted the results in Figure 3.9, where we set the result for one runtime group to 100%.

Groups:	Route 6a:	Route 6b:	Route 39a:	Route 39b:	Route 50a:	Route 50b:	Route 300a:	Route 300b:
1	11.73	11.84	24.49	24.37	11.41	12.18	30.19	26.59
2	11.58	11.58	24.37	24.23	11.29	12.04	30.00	26.38
3	11.44	11.50	24.24	24.16	11.12	11.96	29.92	26.29
4	11.39	11.46	24.14	24.12	11.09	11.93	29.89	26.28
5	11.36	11.44	24.09	24.11	11.06	11.90	29.89	26.28
6	11.33	11.41	24.03	24.10	11.06	11.90	29.87	26.27
7	11.32	11.40	23.97	24.09	11.06	11.88	29.87	26.27
8	11.31	11.40	23.97	24.06	11.05	11.87	29.87	26.27

Table 3.17: Influence of runtime group count on average perceived travel time

What we see, is that the perceived time decreases in the beginning, but decreases significantly slower when using 4 or more runtime groups. We conclude that it is useful to use multiple runtime groups, but using too many of them will not lead to a better perceived travel time. Using about 3 or 4 of them will still lead to a timetable that is easy to remember and is close to optimal with respect to perceived travel time. When using continuous optimization with a linear objective function with non-integral runtimes in combination with average perceived travel time as objective, we get similar results for the determination of runtime groups as in van Oort [2011].

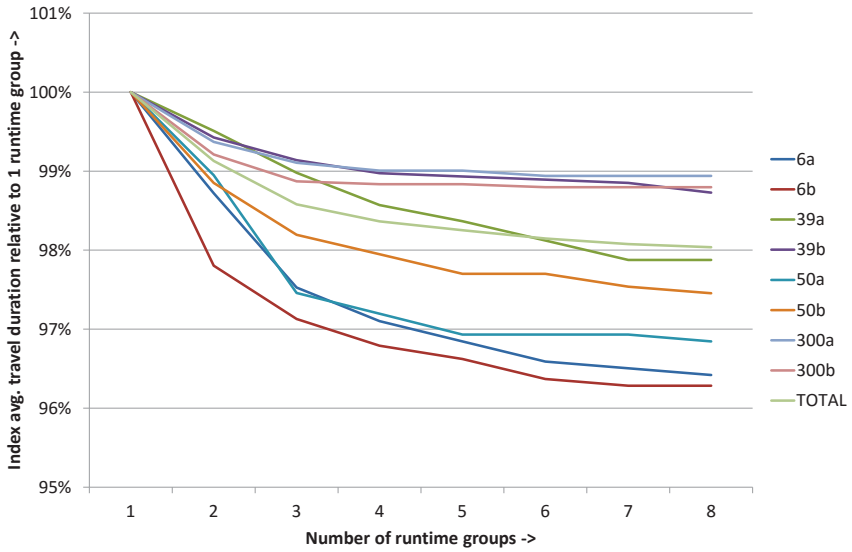


Figure 3.9: Graph showing the relation between average travel time and number of runtime groups. Average travel time with one runtime group is set to 100%.

3.10 General conclusion

In this chapter we have proposed several methods to solve the TRIP RUNTIME DETERMINATION PROBLEM and benchmarked them with the currently widely used percentile-method. After defining the TRIP RUNTIME DETERMINATION PROBLEM in Section 3.3, we have applied several solution methods to solve the TRIP RUNTIME DETERMINATION PROBLEM. Finding an exact solution using INTEGER LINEAR PROGRAMMING turned out to take too long, even for small instances. After relaxing the integrality constraints and moving the decision variables to the objective function, we acquired a result in reasonable time, but at the cost of a possibly non-optimal result.

We have also applied a DYNAMIC PROGRAMMING heuristic to solve the TRIP RUNTIME DETERMINATION PROBLEM. Because the problem characteristics are not fully compliant with the prerequisites of DYNAMIC PROGRAMMING for acquiring the optimal solution, the solution turned out to be suboptimal. Expressing the TRIP RUNTIME DETERMINATION PROBLEM as a continuous optimization problem was the next step. We applied two different cost functions, but in both cases the cost function was not convex, hence we can not prove that the result of this optimization is the global optimum and not a costlier local optimum.

Our conclusion is that in our experiments, continuous optimization with the convex objective function from Equation (3.29) gives the best results in reasonable time. This method gives significantly better results than the widely used percentile method. The average delay can be reduced with about 60% in comparison to the percentile method. Rounding down the resulting timetable times will increase the average delay, but then it will still be the best

method.

What we also saw, was that optimization on passenger travel time instead of on average delay leads to a reduction in travel time of on average 4%, at the cost of punctuality and average delay. When we use a combination of average travel time and average delay as an objective, where both parts have equal weight, we get a reduction of travel time of about 3.5% at the cost of punctuality, but with minor impact on average delay.

When applying the TRIP RUNTIME DETERMINATION PROBLEM to parts of a day, we achieve a better result for the whole day. In experiments we have shown that it is useful to use up to 3 or 4 runtime groups on a day. Using more than 4 runtime groups will not have a significant influence on the result.

Chapter 4

Robust vehicle scheduling

In this chapter we make the step from trip-level optimization to block-level optimization. In the previous chapter, we investigated the data of trips driven in reality, including the actual driving times. Using this information, we determined optimal public trip passing times by tailored methods. In this chapter, we will use the results from these methods for building robust vehicle schedules.

We develop a new method to deal with uncertain trip durations in vehicle scheduling, where we take into account the expected punctuality as well as the expected size of delays. A disruption on a trip will cause a delay on that trip; this is called a *primary delay*. When the delay at the end of a trip is carried over to the next trip, this is called a *secondary delay*.

Our goal is to make vehicle schedules more robust in the sense that secondary delays are avoided as much as possible. Therefore we develop new methods to determine the optimal layovers between trips. We model the trip duration as stochastic variables to reflect the behavior measured in practice. As far as we know, this model is more accurate than the other models we know of, since the used probability function depends on the historic departure times and is based on the measured data. The bottom line is that we will optimize vehicle schedules using traditional techniques in combination with the developed stochastic methods in various ways and compare the expected delay and cost of the results with each other.

In Section 4.1, we describe how vehicle scheduling is traditionally done in the majority of the public transport companies. In Section 4.2, we discuss the different ways that have been described in the literature to take delays into account when making vehicle schedules. We extend the traditional VEHICLE SCHEDULING PROBLEM in Section 4.3 in order to take uncertain trip durations into account.

For the development of our model, we analyze the relation between trip departure time and trip arrival time in reality in Section 4.4. After this analysis, we describe the theoretical model for estimating the trip durations that we will use and for evaluating the punctuality and delays of a vehicle schedule in Section 4.5. Unfortunately, this model can not be implemented directly, so in Section 4.6 we describe how we translate this model into a practical model. A useful model representation is discussed in Section 4.6.1. In Section 4.6.2 we describe how we use this practical model to evaluate punctuality and delays for a new vehicle schedule. Optimization methods based on this knowledge, together with some other methods to optimize a vehicle schedule for robustness are described in Section 4.7.

Computational results of our optimization methods together with an evaluation of the resulting schedules for our optimization model are given in Section 4.8. Our conclusions can be found in Section 4.9.

4.1 Introduction

In the online dictionary by Oxford University Press [2018], one of the definitions of robustness is 'The ability to withstand or overcome adverse conditions or rigorous testing'. In the context of robust vehicle scheduling, we choose to interpret this as 'the resilience of a vehicle schedule to disturbances'. We will base ourselves on secondary delays as a measure for this, as these delays reflect the influence of disturbance of previous trips on the delay at the start of a trip.

As we have seen in Section 1.2, a vehicle schedule is a schedule in which a set of in-service trips is planned to be driven by a set of vehicles. A vehicle schedule contains the mandatory trips, usually the in-service trips from the timetable, plus some necessary pull trips and deadhead trips. The cost of a vehicle schedule is a combination of the fixed cost per vehicle that is needed plus the cost per kilometer or per hour of the vehicles. Fictional costs can be added for objectives that can not be expressed in money, like punctuality. Determining the best vehicle schedule is called the VEHICLE SCHEDULING PROBLEM , which is defined as follows: *Given a set of obligatory trips, determine the vehicle schedule with the lowest cost that contains all these trips.*

To describe the problem properly, we use the following input data. For all trips that are to be scheduled, the planned start and end times are used. For the deadhead trips, a fixed duration is used. A trip may start at the end time of the previous trip, but in most cases there are requirements on the minimum duration of the layover between two trips. The minimum layover is meant to ensure that the trip starts on time, even if the previous trip arrives with a delay. The minimum layover is usually determined by experience and applied for a group of trips. For example, for all trips ending at a station between 6:00 and 19:00, a minimum layover of 5 minutes may be used. In Figure 4.1, a graphical representation of a vehicle schedule with three vehicle tasks is shown.

The VEHICLE SCHEDULING PROBLEM can be solved in numerous ways. In the survey of Bunte and Kliewer [2009], several methods are described and compared. In this chapter, we will only consider the single-depot variant, which can be solved using the network flow model as described in Section 2.4 of Bunte and Kliewer [2009].

In this chapter we will determine ways of minimizing secondary delays in the VEHICLE SCHEDULING PROBLEM, in order to make vehicle schedules more robust.

4.2 Robust Vehicle Scheduling in the literature

As described in Section 4.1, in traditional vehicle scheduling possible delays are taken into account only by using minimum layovers between trips. Primary delays can usually not be avoided by vehicle scheduling, but secondary delays can be influenced by increasing the

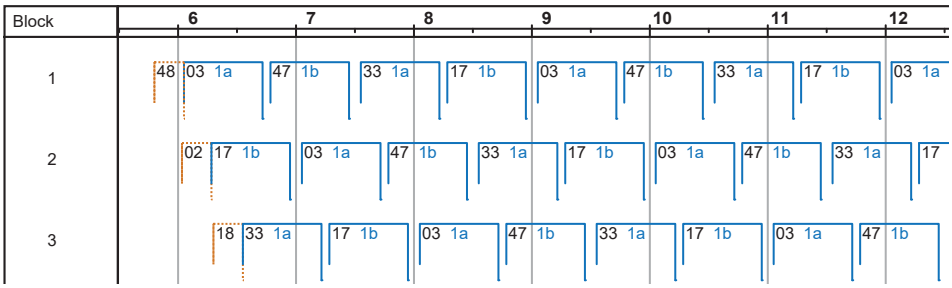


Figure 4.1: Example of Vehicle tasks. On the x-axis, the time of day is denoted. Every horizontal line corresponds to one vehicle task. For example: when we look at the start of the first line, the first trip starts at 5:48. It is dashed and has no route number in it to indicate that it is a deadhead trip. The second trip starts at 6:03 and the text '1a' shows that it is driving route 1 in direction a.

layover between two trips. The longer the layovers, the more robust the vehicle schedule will be.

In Amberg et al. [2011], this way of creating robustness is taken into account while performing vehicle scheduling, crew scheduling and integrated crew- and vehicle scheduling. The authors show that by distributing layovers evenly over the vehicle schedule and increasing layovers, the delay-tolerance of a schedule can be increased at little or no extra cost. In Naumann et al. [2011], vehicle scheduling is solved by Stochastic Programming. The authors use 100 samples of the same network, using real-time trip durations. Fictional cost is added to the objective function for every possible layover value between two trips in the network, representing the secondary delay that is caused when the layover is not long enough to prevent the secondary delay. Delays propagating over more than one arc are not considered; the authors show that increasing the delay cost on arcs by a small amount compensates for this. Using this method, the authors were able to build a more robust vehicle schedule for little or no extra cost.

In Kliewer et al. [2012], the authors allow an extra degree of freedom in order to increase robustness: instead of using a fixed time for every trip, they allow a trip to leave in a predefined time window. In this way, the layovers between trips can be changed in order to reduce secondary delays. However, the duration of a trip is still a fixed value: a trip that starts with a delay, will end with the same delay.

In the field of airline scheduling, a related problem is the assignment of flights to airplanes under uncertainty. This is called Robust Aircraft Routing and is described and studied widely. Some solution methods are described in Chiraphadhanakul and Barnhart [2013], Ben Ahmed et al. [2017], Cadarso and de Celis [2017] and Yan and Kung [2018]. Unfortunately, we can not directly apply these methods to public transport, because some of the methods allow to shift flights, and none of the methods consider the flight duration as a stochastic distribution that is dependent on the departure time.

4.3 Problem description

Our goal is to make the vehicle scheduling problem reflect the reality of transport more accurately. In order to achieve this goal, we deviate from some basic principles used in standard vehicle scheduling in the following ways:

- Trip duration will not be a fixed value, but a stochastic variable, with a distribution that is dependent on the trip departure time.
- Trip departure times and arrival times will be modeled using stochastic variables instead of fixed times.
- We relax constraints on minimum layovers between trips: negative layovers are allowed.
- We allow secondary delays to be absorbed during a trip. Trips that leave too late may arrive in time because the driver typically does not have to wait at holding points.
- We take punctuality during a trip into account.
- We take passenger travel duration and delay into account.

In all literature that we are aware of, the authors assume that a trip takes exactly the planned duration, except in case of a primary delay, when the duration will be longer. In case of a secondary delay, the trip will depart later, but will still have the planned duration. When we take a look at Figure 3.1 on page 29, we see that most trips take less time than planned. Using the widely-used 85th percentile of the actual trip durations for the planned duration, will cause 85 percent of the trips to arrive early. And as we see in Figure 3.1, some trips are always early. Usually, the cause of this is the use of runtime groups instead of determining the optimal planned departure times for every trip. In this chapter, we will take these properties into account.

Deviating from the basic principles used in standard vehicle scheduling has the consequence that traditional vehicle scheduling cannot be used anymore, because departure times, arrival times and trip durations are not a fixed value. In the rest of this chapter, we will propose a method for stochastic vehicle scheduling, which enables us to solve the VEHICLE SCHEDULING PROBLEM with the earlier mentioned, more general principles.

4.4 Observations of trip runtimes

In this section, we will extend the observations from Section 3.2 with some additional observations on trip runtimes and the relation between the start time and end time of a trip. Later in this chapter we will use these observations in order to determine properties of real-life trips that should be implemented in our models.

In Chapter 3, we have studied the trip runtime distribution. We saw that we could decrease the variance of the runtime and increase the punctuality and reliability by using holding points during a trip, where a driver waits until the planned departure time. When we optimized the planned departure times at the stops, we noticed that in order to achieve the highest reliability,

many trips that leave exactly on time will have to wait at one or more holding points during a trip. In Figure 4.2, we depict the data from 10 workdays in October 2012 of route 6 in Groningen (the same we used in Chapter 3) and plot the arrival and departure punctuality at Groningen CS in a graph. We see that the majority of the trips have to wait at 'Groningen Centraal Station', which is a holding point. Arriving later at this stop will for many trips not influence their departure time. This leads to our first observation:

Observation 4. *Starting a trip with a delay may decrease the waiting time at holding points and thus reduce the total actual trip duration.*

Figure 4.2: Graph of arrival and departure punctuality of every trip passing by on route 6 at stop Groningen Centraal Station on 10 workdays in October 2012. Each red dot represents one trip and denotes its arrival and departure punctuality.

When we consider gradually increasing start time delays, we observe that the waiting time at a holding point will decrease, until the point that the trip arrives exactly on time. If

this holds for every holding point in this trip, the actual total trip duration will not change anymore. When this holds for every trip, the distribution of trip runtime duration will not change. This leads to:

Observation 5. *The actual trip runtime distribution is dependent on the delay at the start of the trip.*

This observation also holds for the distribution of actual passing times at stops. From these passing times, we can calculate the expected delay cost and passenger travel time for every departure time. This cost is expressed as a function of the actual departure time.

From these observations we conclude that a new model should use a runtime distribution that is dependent on the actual departure time. This will be used in the next section.

4.5 Theoretical model of STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM

In this section, we will develop a theoretical model for the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM (SDTDVSP) using the observations from Section 4.4. We will first define the punctuality cost of a trip, along with some more prerequisites. We will end this section with our definition of the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM.

Definition 1. *We denote the probability that trip i leaving at t_{di} arrives at t_{ai} by the function $p_i(t_{ai}|t_{di})$, where $t_{di}, t_{ai} \in \mathbb{R}$.*

Because for a given t_{di} the function $p_i(t_{ai}|t_{di})$ is the probability density function (PDF) of the arrival time of trip i leaving at t_{di}, we have that for all trips i and every departure time t_{di}:

$$\int_{t_{ai} \in \mathbb{R}} p_i(t_{ai}|t_{di}) \, \mathrm{d}t_{ai} = 1 \tag{4.1}$$

Definition 2. *Let the PDF $q_i(t)$ be the distribution of the departure time of trip i.*

The PDF of the arrival time $r_i(t)$ at the end of trip i can be calculated from $p_i(t_{ai}|t_{di})$ and $q_i(t)$ as follows:

$$r_i(t) = \int_{s \in \mathbb{R}} p_i(t|s) q_i(s) \, \mathrm{d}s \tag{4.2}$$

The punctuality cost of a trip can be defined in several ways, but should by observation always depend on the departure time of a trip. For example, we could assign a cost of 1 when the trip arrives too late, and 0 otherwise. Or we could assign the total delay of all passengers on this trip. We use $c_i(t_{di})$ to denote the punctuality cost for trip i when it departs at t_{di}. When the departure time is distributed as in the previous formula and denoted as $r_i(t)$, the expected punctuality cost of trip i can be calculated by:

$$C_i = \int_{t \in \mathbb{R}} c_i(t) r_i(t) \, \mathrm{d}t \tag{4.3}$$

For the function $p_i(t_{ai}|t_{di})$, there are two special cases of t_{di} left to describe:

- Trip i has a departure time t_{di} that is before the planned departure time $t_{d(planned)i}$. In this case the driver waits until the planned departure time:

$$\forall t_{di} \leq t_{d(planned)i} : p_i(t_{ai}|t_{di}) = p_i(t_{ai}|t_{d(planned)i}) \tag{4.4}$$

- Trip i leaves after the earliest departure time where the driver does not have to wait anywhere; this time is denoted by $t_{d(nowait)i}$. The trip duration distribution will be equal to that at departure time $t_{d(nowait)i}$:

$$\forall t_{di} \geq t_{d(nowait)i} : p_i(t_{ai}|t_{di}) = p_i(t_{ai} - (t_{di} - t_{d(nowait)i})|t_{d(nowait)i}) \tag{4.5}$$

For deadhead trips, there are no intermediate stops and no values for $d(planned)$ and $d(nowait)$, so Equations (4.4) and (4.5) are not applicable.

A vehicle schedule consists of trip sequences that are to be driven consecutively by one vehicle. For every trip i in such a vehicle task, $p_i(t_{ai}|t_{di})$ and $c_i(t_{di})$ are given. We start with a given PDF $q_i(t)$ for the departure time of the first trip of the vehicle task. Then for every trip in the vehicle task, we apply Equation (4.2) to get the arrival time distribution after every trip. For every trip, we also calculate the cost for the trip using Equation (4.3). The cost of a vehicle task is the total cost of all trips in a vehicle task, optionally increased with the fixed cost of one vehicle. The sum of all the vehicle task costs is the cost of the vehicle schedule.

We now define the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM as follows:

Definition 3. *Let a set of service trips be given, with for each trip a planned departure time and a stochastic duration which is dependent on the actual departure time. Furthermore, for every trip the cost is given as a function of the actual departure time. The* STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM *is defined as the problem of finding the vehicle schedule with minimal expected cost.*

4.6 Discretized model for solving the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM

The variables t_{di}, t_{ai} and functions $p_i(t_{ai}|t_{di})$, $q_i(t_{di})$, $r_i(t_{di})$ and $c_i(t_{di})$ use and yield continuous values. Because of the complex nature of trips, it is very likely that these functions and the time distributions can not be evaluated analytically, so we have to use some sort of approximation. We will do this by using discrete times for t_{di} and t_{ai}, where we will use a granularity of one minute. We use one minute because in the majority of the public transport in the Netherlands and abroad, planning is done in minutes; note that other values for the granularity may be chosen as well. Using discrete times, we will use row vectors to reflect probability distributions of times r_i, q_i and c_i and transition matrices to reflect stochastic trip durations.

A stochastic time t in our algorithm is described by its discrete PDF. In our approximation we represent t by a row vector with one element for every possible time, containing the probability for that time. The sum of all elements has to be 1. However, many of the elements will be zero, especially those at the tails of the vector. For example: a stochastic time t which is in 4 percent of the cases 4, in 90 percent of the cases 5 and in the remaining 6 percent of the cases 6 gives the vector given in (4.6).

$$t = \begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.04 & 0.90 & 0.06 & 0.00 & 0.00 & 0.00 \end{pmatrix} \end{array} \tag{4.6}$$

The duration of a trip can be modeled by taking the distribution of the arrival time for every departure time and putting that in a matrix. We call this the *transition matrix* T_i of a trip i. Every row corresponds to a different departure time t_{di}, in which the distribution of arrival time t_{ai} is given. Every cell is the probability that, given a departure time t_{di}, the trip arrives at exactly t_{ai}. In fact, this matrix contains the numbers $p(t_{ai}|t_{di})$. An example of a matrix T_i can be found in (4.7).

$$T_i = \begin{array}{c} \begin{array}{ccccccc} t_{ai}=6 & t_{ai}=7 & t_{ai}=8 & t_{ai}=9 & t_{ai}=10 & t_{ai}=11 & t_{ai}=12 \end{array} \\ \begin{pmatrix} 0.00 & 0.60 & 0.35 & 0.05 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.60 & 0.35 & 0.05 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.60 & 0.35 & 0.05 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.60 & 0.35 & 0.05 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.30 & 0.55 & 0.15 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.15 & 0.40 & 0.45 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.15 & 0.40 & 0.45 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.15 & 0.40 & 0.45 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.15 & 0.40 & 0.45 \end{pmatrix} \end{array} \begin{array}{l} t_{di}=1 \\ t_{di}=2 \\ t_{di}=3 \\ t_{di}=4 \\ t_{di}=5 \\ t_{di}=6 \\ t_{di}=7 \\ t_{di}=8 \\ t_{di}=9 \end{array} \tag{4.7}$$

In our algorithm, we use the following variables:

- t : Row vector which reflects a time. Every element in the vector represents the probability for that time.
- C : Variable which keeps track of the total cost of the vehicle task.
- T_i : Transition matrix that represents the characteristics of the trip duration of trip i.
- c_i : Column vector that defines the cost of trip i depending on the departure time.

Furthermore, we use the discretized version of Equation (4.2):

$$r_i(t_{ai}) = \sum_{t_{ai},t_{di}} q(t_{di}) p_i(t_{ai}|t_{di}) = t \cdot T_i \tag{4.8}$$

And we use the discretized version of Equation (4.3):

$$c(t_{ai}) = \sum_{t_{di}} q(t_{di}) C_i(t_{di}) \tag{4.9}$$

The variables can be used to evaluate a vehicle task as described in Algorithm 2.

```
1 Data: Matrices T_i and C_i for every trip i
2 Result: Punctuality-related cost of a vehicle task
3 C ← 0 ;
4 t ← (1  0  0  ...);
5 foreach trip i in vehicle task do
6 |   C ← C + t c_i ;
7 |   t ← t T_i ;
8 end
```

Algorithm 2: Vehicle task evaluation with c representing the cost of the vehicle task and t reflecting the current time

After an initialization of total cost C to 0 in line 3 and stochastic time t to be always 0 in line 4 of Algorithm 2, we perform the following two calculations for every trip i in the vehicle task:

- In line 6, we calculate the cost of trip i by multiplying the current stochastic time t with cost vector c_i. The result is added to total cost C. The calculation in this step is according to Equation (4.9).
- In line 7, the stochastic time t is updated for performing trip i by multiplying the vector t with transition matrix T_i. The calculation in this step is according to Equation (4.8).

4.6.1 Compact representation of the information in the matrices

When implementing Algorithm 2, we see that the matrices and vectors used are large and mainly filled with zeroes. This will lead to unnecessary long running times. In this section we will discuss these issues and the speed-ups we made in the implementation.

Let us look again at the example time vector in Equation (4.6). We will only store the contiguous part that contains all non-zeroes and store the index idx of the first element with it. We denote this partial vector as $t = (.)_{idx}$. For example, the time vector from Equation (4.6) will become

$$t = \begin{pmatrix} 0.04 & 0.90 & 0.06 \end{pmatrix}_4 \tag{4.10}$$

In the example in Equation (4.7) we see an example matrix T_i for trip i. We note that the first four rows are identical, since an early departure does not change the distribution of the arrival time. This happens because vehicles wait for the planned departure time at the start of the trip. This behavior is also described in Equation (4.4). The last four rows are also identical, except for a shift by one minute in each row. This means that after $d = 6$, the distribution of the arrival time of a trip shifts along with the departure time. Such a behavior occurs when a bus does not have to wait for the departure time anywhere; this is also described in Equation (4.5)

These patterns typically occur at every trip. Hence we can compress the matrix further, as long as we take these patterns into account when using the compressed matrices for calculations. Like the time vector t, the compressed matrix also has a base time for the

departure time and for the arrival time. These base times do not have to be equal. We denote a matrix with departure base j and arrival base k as $T_i = \left(.\right)_{j,k}$. All elements in a row always have a sum of 1.

For the example matrix T_i, we use $d = 4$ as departure base time, because in the rows with $d < 4$, the rows are identical to the row with $d = 4$. We only include the rows until $d = 6$, after which the contents of the row with $d = 6$ are repeated. For the columns, we use the base $a = 7$ and use the rows until $a = 9$, because the rest is zero or a repetition of the bottom row. Matrix (4.7) will thus be written as

$$T_i = \begin{pmatrix} 0.60 & 0.35 & 0.05 \\ 0.30 & 0.55 & 0.15 \\ 0.15 & 0.40 & 0.45 \end{pmatrix}_{4,7} \tag{4.11}$$

Similarly, we use a compressed matrix C_i to calculate the cost related to vehicle, crew, punctuality and passenger travel times of trip i. Every element of C_i corresponds to the cost when a vehicle is ready to drive a certain trip. When the vehicle is too early, it waits for the planned departure time, otherwise it leaves too late. So for early vehicles, the crew and vehicle cost associated to waiting for departure are also included. When we multiply this matrix with the departure time matrix, we get the cost of this trip. An example of a cost matrix, with a base of 4 is the following:

$$C_i = \begin{pmatrix} 80 \\ 120 \\ 200 \end{pmatrix}_4 \tag{4.12}$$

For deadhead trips, we will use similar matrices. Because deadheads do not have a fixed starting time but only a runtime distribution, we will use a matrix T with only one row and with base 0.

4.6.2 Evaluation of vehicle tasks

A vehicle task consists of a sequence of trips to be driven in a given order. For every trip i, we know the matrices T_i and C_i. For the departure time of the vehicle task, we know the distribution of the departure time. When we would have all full matrices, not compressed as in Section 4.6.1, we can use Algorithm 2. When we use compressed matrices, we will use Algorithm 3, which is explained below.

Data: Matrices T_i and C_i for every trip i
Result: Punctuality-related cost of a vehicle task
$c \leftarrow 0$;
$t \leftarrow \begin{pmatrix}1 & 0 & 0 & \dots\end{pmatrix}$;
foreach *trip* i *in vehicle task* **do**
    Change base of t, C_i and departure base of T_i to the minimum of the base of t, the base of C_i and the base of T_i ;
    $c \leftarrow c + t\ C_i$;
    $t \leftarrow t\ T_i$;
end

Algorithm 3: Vehicle task evaluation with compressed matrices

In the algorithm, it is necessary to change the base of matrices or vectors, in order to be able to use them in calculations. For example, we have a matrix T_1 with base time b_1 which we want to convert to a matrix T_2 with base time b_2 where $b_1 \geq b_2$ (this is a consequence of starting with a base value equal to the minimum value of multiple bases). Obviously, when $b_1 = b_2$, nothing has to be done. When $b_1 > b_2$, we add $b_1 - b_2$ extra rows which are duplicates of the original first row for matrices. Examples of rebasing t, T_i and C_i from base 4 to base 2 are given in (4.13), (4.14) and (4.15).

$$t = \begin{pmatrix}0.04 & 0.90 & 0.06\end{pmatrix}_4 = \begin{pmatrix}0.00 & 0.00 & 0.04 & 0.90 & 0.06\end{pmatrix}_2 \tag{4.13}$$

$$T_i = \begin{pmatrix}0.60 & 0.35 & 0.05\\ 0.30 & 0.55 & 0.15\\ 0.15 & 0.40 & 0.45\end{pmatrix}_{4,7} = \begin{pmatrix}0.60 & 0.35 & 0.05\\ 0.60 & 0.35 & 0.05\\ 0.60 & 0.35 & 0.05\\ 0.30 & 0.55 & 0.15\\ 0.15 & 0.40 & 0.45\end{pmatrix}_{2,7} \tag{4.14}$$

$$C_i = \begin{pmatrix}80\\ 120\\ 200\end{pmatrix}_4 = \begin{pmatrix}80\\ 80\\ 80\\ 120\\ 200\end{pmatrix}_2 \tag{4.15}$$

4.7 Solving the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM

In Section 4.6.2, we described how we evaluate vehicle tasks for cost taking the stochastic nature of trips into account. This measure is the starting point for the optimizations we will do and every outcome on vehicle scheduling in this chapter will be evaluated in this way.

We propose a new approach to find an optimal robust vehicle schedule. For solving the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM we model

the problem as a graph, in which every trip is a node and where every allowed connection between two trips is represented by an arc. The depot is represented by two nodes, one at the start of the day where the vehicles start and one at the end of the day. Costs are assigned to the arcs. The objective is to find a set of paths from depot to depot in which all trips are included once and for which the total cost is minimal. This is done by formulating the problem as an INTEGER LINEAR PROGRAMMING problem and solving this using CPLEX. In Section 4.7.1, we propose methods to implement the robustness factor by changing the arcs that connect two trips that may be driven sequentially. Optimization will be done using the earlier mentioned matrices in Section 4.7.2.

4.7.1 Adjusting layover arcs

Traditionally, the minimum layover after a trip is determined before optimization, and arcs that represent a layover that is too short are simply not generated. In our optimizations, we will generate these arcs anyway, but we will assign an extra cost to every arc to model the effect on punctuality. An arc representing a long layover will get a lower cost than an arc representing a short layover which may cause delay propagation much more easily.

The idea of changing the cost of arcs in order to make a vehicle schedule more robust, is not new. It has been done earlier in Naumann et al. [2011], where the cost is proportional to the average delayed departure of the second trip. However, we take this idea a step further in two ways:

- We add extra cost to an arc, which is the extra cost of the secondary delays of the second trip given the fact that the first trip leaves exactly on time. We use the historic measurements to determine the distribution of the duration of the first trip. The closer two trips are together, the larger the influence of the first trip will be, and hence the extra cost of the arc will be larger. For the cost of the delay of the second trip, we look at a combination of the average start delay caused by the first trip and the average end delay. This end delay is determined by the method in Section 4.7.2. For example: if trip 1 ends at 9:01, then the arc to trip 2 planned to leave at 9:02 will have an extra cost compared to trip 3 planned to leave at 9:04 because trip 2 is more prone to delay than trip 3.
- We allow *negative layovers*. Usually, overlapping trips are not allowed in a vehicle task. We observed earlier that most trips arrive too early, and we observed that trips that leave too late will arrive with a delay lower than the delay at departure, i.e. the delay is reduced during driving. Both observations make it likely that negative layovers between trips could be useful and acceptable. When using negative layovers, we still require the second trip to start later than the start of the first trip to avoid cycles in our graph. For example, if trip 1 ends at 9:01, we allow an arc to a trip 2 leaving at 8:59.

4.7.2 Using transition matrices over two trips

In Section 4.7.1, we adjusted layover arc costs by only considering two consecutive trips, where the first trip is considered to leave exactly on time and the departure delay of the second

trip is taken into account. In practice, it may occur that a delay in one trip has effect on more than one trip in the vehicle task. When creating optimal vehicle schedules, we should take this observation into account.

The next step is to also take the arrival delay of the second trip into account. To achieve this, we apply the transition matrix from Section 4.6.1. We start with an evaluation of the possible savings when we take delay propagation over more than one trip into account. The properties of the second trip are determined by evaluating the sequence of the two trips on either side of the arc using the method described earlier in Section 4.6.2, assuming that the first trip leaves on time. We use different cost calculations and compare these, see Section 4.8.1. The effect of increasing this cost on the solution will give an indication of the possible effect of taking delay propagation on longer sequences of trips into account. If the solution will be more expensive when increasing this cost, we know that delays will propagate over more than one trip, and that we should take this into account. Solving the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM while taking delay propagation on sequences of 3 or more trips into account is a subject for future research.

4.8 Computational results

In this section we will compare the methods for determining of robust vehicle schedules with each other and with the methods that are currently widely used. The methods and the way they are used are described in Section 4.8.1. The data we used in this comparison and the way we handle these data is described in Section 4.8.2.

4.8.1 The methods used in the comparison

In order to test our robust vehicle scheduling methods from Section 4.7, we compared five traditional methods of vehicle scheduling with some variants of the method of vehicle scheduling described in Section 4.7.1, in which we vary the cost of layovers according to expected punctuality. The methods that we use in our tests are mentioned below. The five traditional methods only adjust the minimal layover duration and hence the arcs in the graph:

- *Standard*. Traditional vehicle scheduling, using the minimum layovers that are currently in use in practice.
- *MinLay fixed, Time*. Traditional vehicle scheduling, with a fixed minimum layover after every trip, for every trip the same value.
- *MinLay fixed, Percentage*. Traditional vehicle scheduling, with a minimum layover that is a fixed percentage of the trip duration.
- *Percentile, MinLay*≥ 0. Traditional vehicle scheduling, using a minimum layover that is a percentile-value of measured trip durations minus the planned trip durations and that is non-negative.

- *Percentile*. Traditional vehicle scheduling, using a minimum layover that is a percentile-value of measured trip durations minus the planned trip duration, also allowing negative layovers.

The next few methods use the start delay of the successor trip for determination of the layover arc cost, the minimal layover duration is implemented by adjusting the graph:

- *StartPunctLin, MinLay*≥ 0. Robust vehicle scheduling as described in Section 4.7.2, applying an extra cost to a layover that is a factor LCF times the probability that the second trip leaves more than 0 minutes too late. Negative layovers are not allowed.
- *StartPunctLin*. Robust vehicle scheduling, applying an extra cost to a layover that is a factor LCF times the probability that the second trip leaves more than 0 minutes too late, also allowing negative layovers.
- *LinearStartDelay*. Robust vehicle scheduling, applying an extra cost to a layover that is a factor $LinS$ times the average delay in minutes at start of the second trip, also allowing negative layovers.
- *SquaredStartDelay*. Robust vehicle scheduling, applying an extra cost to a layover that is a factor SqS times the square of the average delay at start of the second trip, also allowing negative layovers.

The last method in our list uses the start and end delay of the successor trip for determination of the layover arc cost, the minimal layover duration is implemented by adjusting the graph:

- *LinSD+LinED*. Robust vehicle scheduling applying an extra cost to a layover that is a the sum of a factor $LinS$ times the average delay in minutes at start of the second trip and a factor $LinE$ times the average delay in minutes at end of the second trip.

The traditional methods and the percentile methods determine the minimum layover before optimization and remove arcs that represent layovers that are too short. The rest of the methods assign extra cost to layovers. The last method is mainly used for evaluating the possible effect of having multi-trip delay propagation as described in Section 4.7.2. An overview of the methods and their properties can be found in Table 4.1.

Method	Duration minimum layover	Layover cost proportional to	Neg. layover allowed
Standard	Fixed, same values as used in practice	-	No
MinLay fixed Time	Same value for all trips	-	No
MinLay fixed Percentage	Percentage of planned trip duration	-	No
Percentile, MinLay>0	Percentile value of measured trip duration	-	No
Percentile	Percentile value of measured trip duration	-	Yes
StartPunctLin, MinLay>0	-	Nr. of late departures trip after layover	No
StartPunctLin	-	Nr. of late departures trip after layover	Yes
LinearStartDelay	-	Avg. Delay at Start of trip after layover	Yes
SquaredStartDelay	-	Avg. Delay at Start of trip after layover	Yes
LinSD+LinED	-	Avg. Delay at Start and End of trip after layover	Yes

Table 4.1: Properties of the optimization methods used for solving the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM.

4.8.2 The data used in the comparison

The data we used for our comparison is provided by Qbuzz, a public transport organization in the Netherlands. The dataset contains very detailed measurements of the public transport by bus in and around the city of Utrecht for all weekdays in September, October and November of 2014 for 3496 trips per day. For every day, for every trip, for every stop we know the arrival and departure time, and the number of passengers that enter and leave the bus. From this data, we calculate the matrices and cost vectors for every trip according to the following steps:

- First we remove the extra loading time that is spent at holding points where the driver has to wait for the planned departure time. As we have seen in Section 3.2, we can do this by bounding the loading time at holding points to 45 seconds.
- For every delayed departure with a delay from 0 to 10 minutes, we calculate the expected arrival and departure times for every stop of every measured trip in the dataset. These expected times are calculated using the measurements, adding extra time when a driver has to wait at a holding point.
- For every delayed departure, we determine a statistical distribution for the runtimes. If there are fewer than 10 measurements, we smoothen the distribution by fitting a shifted Gamma distribution to the runtime measurements. When there are 10 or more measurements, we use an empirical distribution using the measurements.
- Fill the matrix for this trip using this distribution.
- Calculate the cost for every delay and put this information into the cost vector.

For fitting a shifted Gamma distribution with location parameter $\tilde{\gamma}$, shape parameter $\tilde{\alpha}$ and rate parameter $\tilde{\beta}$, we start with determining the shift parameter using the procedure described in Law [2007] in Section 6.8. For the sorted set of measurements $X_1..X_n$, where X_1 is the smallest measurement, we estimate the location parameter $\tilde{\gamma}$ using:

$$\tilde{\gamma} = \frac{X_1 X_n - X_k^2}{X_1 + X_n - 2X_k} \tag{4.16}$$

where k is the smallest integer in $\{2, 3, ..., n-1\}$ such that $X_k > X_1$. Then we transform the measurements X into a set X' by

$$X'_i = X_i - \tilde{\gamma}, \forall i = 1, .., n \tag{4.17}$$

The parameters $\tilde{\alpha}$ and $\tilde{\beta}$ are estimated using Maximum Likelihood Estimation. They are calculated by numerically solving

$$\ln \tilde{\beta} + \Psi(\tilde{\alpha}) = \frac{\sum_{i=1}^{n} \ln X'_i}{n}, \tilde{\alpha}\tilde{\beta} = \frac{\sum_{i=1}^{n} X'_i}{n} \tag{4.18}$$

where $\Psi(\tilde{\alpha})$ is the digamma function, defined by $\Psi(\tilde{\alpha}) = \frac{\Gamma'(\tilde{\alpha})}{\Gamma(\tilde{\alpha})}$.

For deadhead trips and for trips without measurements, we assume that the driving time follows a normal distribution with a standard deviation σ of 10 percent of the planned duration, and a planned duration of the mean plus standard deviation, $\mu + \sigma$.

4.8.3 Results of the comparison

For our experiments we have used the timetable and all runtime measurements for the urban lines of Utrecht in September, October and November of 2014. For every optimization method in Section 4.8.1, we have run the test several times with different values for the parameters. We have evaluated the resulting schedules from the optimization using the transition matrices as described in Section 4.6.2, which is in essence a way of doing many simulation runs simultaneously. For every test we obtain the following figures, which are shown in Table 4.2:

- *Method used* – Which method from the list in Section 4.8.1
- *Parameters* – The parameters that are applied in this method
- *Cost* – Total cost of the resulting schedule
- *Departure punctuality* – Percentage of trips that leave in time
- *Average Delay at start* – Average delay of all trips at departure
- *Average Delay at end* – Average delay of all trips at arrival
- *Number of vehicles* – Number of vehicles necessary for this schedule
- *In-service time* – Total in-service time of this schedule
- *Deadhead time* – Total deadhead time of this schedule
- *Stationary time* – Total stationary time of this schedule
- *Total time* – Total time of this schedule

In order to make the results from Table 4.2 more comprehensible, we have plotted the results in Graph 4.3, horizontally the cost per day and vertically the start punctuality. The results are also plotted in Graph 4.4. Here we put the average start delay on the vertical axis.

When we look at the results in Table 4.2 and in Figures 4.3 and 4.4, we notice the following:

- All methods with a predetermined minimum layover perform less than the models that add a fictional cost to the layover representing the punctuality. In Figure 4.3 we see that for the same cost, the punctuality is about two to three percent better and in Figure 4.4 we see that for the same cost, the average delay is reduced with 30 to 40 percent.
- Relaxing the constraint that a layover always has to be positive increases the punctuality by one percent at the cheaper and least robust scenarios. Because these scenarios are generally not desirable, it is not useful to relaxing this constraint.
- When comparing the results from the methods LinearStartDelay, LinSD+LinED2 and LinSD+LinED5, we see that putting an extra cost on the end delay as described in Section 4.7.2 will not increase the average delay when having a schedule for the same cost, and will decrease the punctuality by a small number. We conclude that taking into account delay propagation over longer sequences of trips will not have a huge benefit.

Method used	Parameters	Cost	Departure Punct	Avg Dly Start	Avg Dly End	Vehicles	InSrv	Dhd	Stat	Tot
Standard		111898	87.5	0.62	1.51	137	1526:00	109:08	256:58	1892:06
MinLay fixed, Time	0 min	109458	82.5	0.92	1.72	134	1526:00	106:39	204:18	1836:57
MinLay fixed, Time	1 min	114652	90.9	0.47	1.38	142	1526:00	114:51	308:46	1949:37
MinLay fixed, Time	2 min	116439	93.0	0.39	1.33	145	1526:00	121:45	341:11	1988:56
MinLay fixed, Time	3 min	122285	96.3	0.23	1.23	155	1526:00	132:55	448:35	2107:30
MinLay fixed, Time	4 min	124634	97.1	0.20	1.21	158	1526:00	137:07	493:05	2156:12
MinLay fixed, Time	5 min	126824	97.7	0.17	1.19	163	1526:00	137:37	533:01	2196:38
MinLay fixed, Time	7 min	133009	98.5	0.12	1.15	172	1526:00	150:56	638:58	2315:54
MinLay fixed, Time	10 min	140333	99.2	0.07	1.12	182	1526:00	157:45	802:42	2486:27
MinLay fixed, Percentage	5 percent	113322	88.3	0.58	1.47	140	1526:00	118:27	272:45	1917:12
MinLay fixed, Percentage	10 percent	117853	93.4	0.35	1.31	147	1526:00	122:31	362:44	2011:15
MinLay fixed, Percentage	15 percent	122603	95.8	0.23	1.23	156	1526:00	133:21	446:53	2106:14
MinLay fixed, Percentage	20 percent	127373	97.6	0.16	1.18	165	1526:00	143:10	537:07	2206:17
Percentile, MinLay>0	pct=50	112693	87.5	0.59	1.46	140	1526:00	114:56	246:10	1887:06
Percentile, MinLay>0	pct=65	115167	90.2	0.45	1.37	145	1526:00	123:59	274:18	1924:17
Percentile, MinLay>0	pct=85	121561	95.8	0.22	1.22	159	1526:00	138:29	367:15	2031:44
Percentile, MinLay>0	pct=95	130308	98.6	0.10	1.15	176	1526:00	160:23	491:55	2178:18
Percentile	pct=50	111804	83.9	0.70	1.52	140	1526:00	119:40	212:01	1857:41
Percentile	pct=65	114739	88.9	0.50	1.40	145	1526:00	124:20	257:23	1907:43
Percentile	pct=85	121479	95.7	0.22	1.22	159	1526:00	138:05	365:38	2029:43
Percentile	pct=95	130302	98.6	0.10	1.14	176	1526:00	159:59	492:04	2178:03
StartPunctLin, MinLay>0	LCF=1	110762	87.8	0.55	1.43	135	1526:00	111:02	232:36	1869:38
StartPunctLin, MinLay>0	LCF=2	111004	88.7	0.51	1.41	135	1526:00	110:44	241:14	1877:58
StartPunctLin, MinLay>0	LCF=5	113141	91.1	0.38	1.32	138	1526:00	113:58	290:01	1929:59
StartPunctLin, MinLay>0	LCF=10	117352	95.5	0.22	1.21	144	1526:00	121:48	363:15	2011:03
StartPunctLin, MinLay>0	LCF=20	122824	97.8	0.11	1.14	153	1526:00	139:56	434:32	2100:28
StartPunctLin, MinLay>0	LCF=50	130655	99.0	0.04	1.09	163	1526:00	180:24	527:25	2233:49
StartPunctLin	LCF=2	101463	65.0	2.48	2.90	116	1526:00	92:31	101:07	1719:38
StartPunctLin	LCF=3	105956	79.8	1.08	1.82	125	1526:00	96:17	172:04	1794:21
StartPunctLin	LCF=4	107625	84.3	0.74	1.57	128	1526:00	98:52	205:55	1830:47
StartPunctLin	LCF=5	109250	87.5	0.58	1.46	131	1526:00	99:39	235:51	1861:30
StartPunctLin	LCF=10	113976	93.6	0.31	1.28	139	1526:00	105:52	330:25	1962:17
StartPunctLin	LCF=20	120150	97.3	0.15	1.17	150	1526:00	117:44	427:05	2070:49
StartPunctLin	LCF=50	127403	98.8	0.06	1.11	160	1526:00	142:02	550:28	2218:30
LinearStartDelay	LinS = 5	107500	83.8	0.75	1.57	128	1526:00	99:37	199:53	1825:30
LinearStartDelay	LinS = 10	111761	91.1	0.40	1.34	135	1526:00	102:56	285:29	1914:25
LinearStartDelay	LinS = 20	118590	96.5	0.17	1.19	147	1526:00	114:14	400:04	2040:18
SquaredStartDelay	SqS = 5	110649	88.2	0.50	1.39	136	1526:00	107:05	238:26	1871:31
SquaredStartDelay	SqS = 10	113807	91.7	0.34	1.29	141	1526:00	115:07	285:00	1926:07
SquaredStartDelay	SqS = 20	117747	95.0	0.21	1.20	148	1526:00	122:15	351:48	2000:03
LinSD+LinED2	LinS/LinE=5/2	108712	86.1	0.63	1.48	131	1526:00	103:23	214:01	1843:24
LinSD+LinED2	LinS/LinE=10/2	112703	92.0	0.36	1.31	137	1526:00	106:07	300:29	1932:36
LinSD+LinED2	LinS/LinE=20/2	119231	96.7	0.16	1.17	148	1526:00	117:44	402:07	2045:51
LinSD+LinED5	LinS/LinE=5/5	110767	89.0	0.48	1.38	134	1526:00	110:11	246:04	1882:15
LinSD+LinED5	LinS/LinE=10/5	115193	93.5	0.28	1.25	141	1526:00	118:30	323:54	1968:24
LinSD+LinED5	LinS/LinE=20/5	120589	96.8	0.14	1.16	149	1526:00	133:01	400:11	2059:12

Table 4.2: Results for the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM

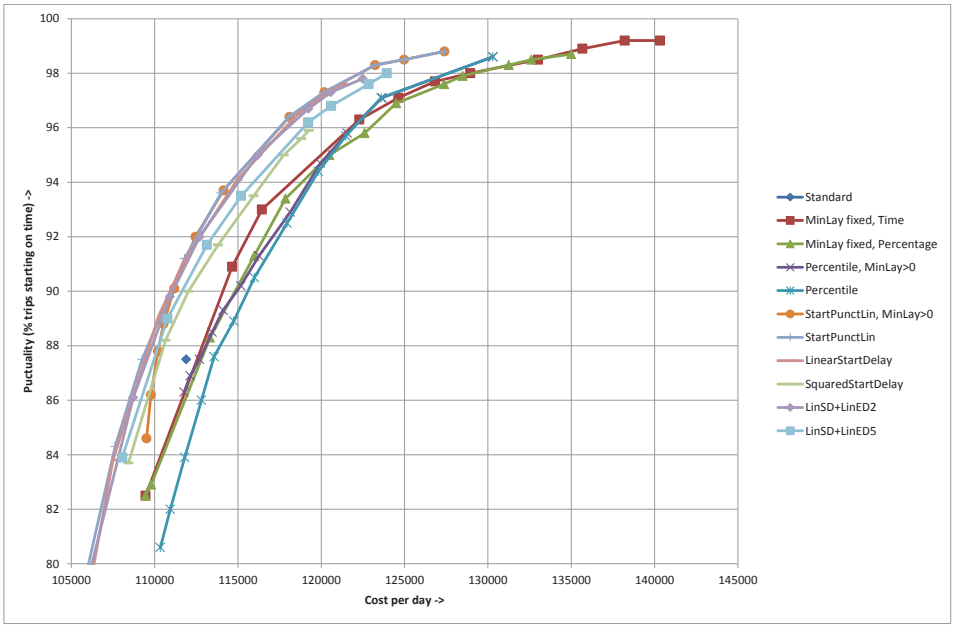


Figure 4.3: Graph of the results of all experiments for the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM. In order to be able to compare the results, we have compared the punctuality (percentage of trips that leave on time) with the cost per day.

4.9 General conclusion

In this chapter we have investigated how one may create more robust vehicle schedules for public transport. After a thorough analysis of trip runtime measurements in Section 4.4, we translated our observations in a theoretical model in Section 4.5. This theoretical model made it possible to evaluate a vehicle schedule for punctuality, but because all functions and distributions are continuous, it was not usable in practice. In Section 4.6 we discretized these functions and distributions in order to make evaluation of vehicle schedules possible.

In Section 4.7 we described a way to optimize a vehicle schedule based on the evaluation method we described earlier in Section 4.6. Where most current methods implement robustness by setting a minimum layover duration, we did not impose boundaries to layover duration. Instead, we assigned a fictional cost to each layover dependent on the punctuality of the trip after the layover. We tested with several combinations of start and end punctuality of the second trip.

When we compare the results, we see that the methods using a fictional cost assigned to the layovers performs better than limiting the layover duration beforehand. We see that for the same cost, the punctuality is about 2 to 3 percent better and the average delay is reduced with 30 to 40 percent. Which method of calculation of the fictional layover cost is used does

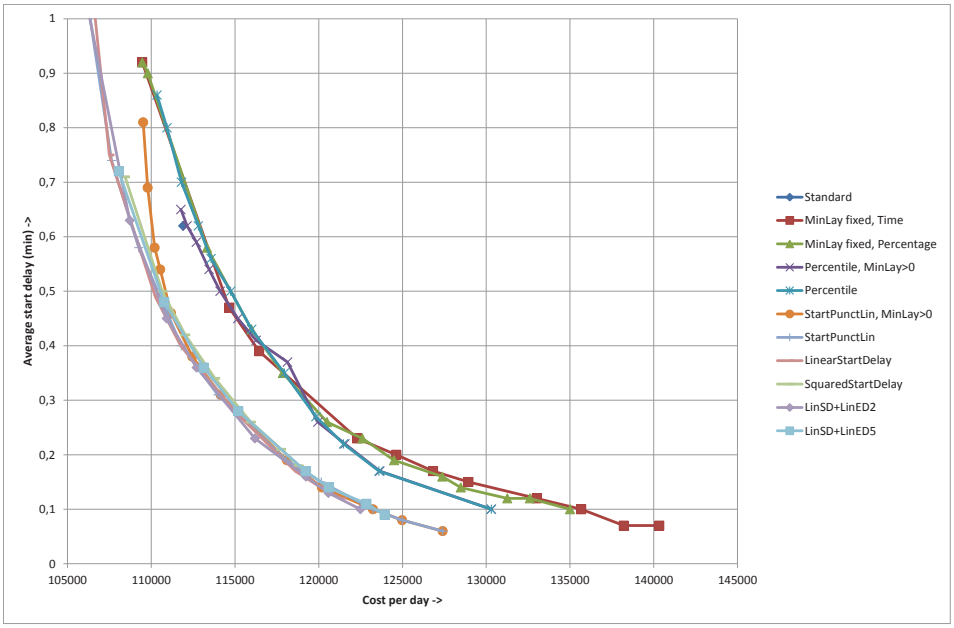


Figure 4.4: Graph of the results of all experiments for the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM. In order to be able to compare the results, we have compared the average start delay with the cost per day.

not have a significant influence on the results. The same holds for the method of determining the minimum layover duration.

For calculation of the fictional layover cost, we only evaluated the punctuality of at most two consecutive trips. We concluded that for our dataset, taking longer sequences into account is not useful. However, in other timetables we do not know if using longer sequences is useful. For this, future research must be done on taking longer sequences of trips into account.

Chapter 5

Scheduling electric vehicles in public transport[1]

5.1 Introduction

In the last years, the trend is that public transport becomes more and more environmentally friendly. European norms for engine exhaust gases have become stricter over time, as we can read on the website `http://www.dieselnet.com/standards/eu/hd.php` of the ECOpoint [2018]. Therefore, new inventions like hybrid and fully electric powered vehicles are now introduced.

For electric vehicles (EVs), there are a few pilot projects in the Netherlands. Small fleets of EVs currently drive in Utrecht, Dordrecht, Gorinchem, Groningen and Eindhoven. For a large-scale extension of the use of EVs, a lot of problems have to be solved. One of them is vehicle scheduling. This is mainly because currently batteries in EVs do not have enough capacity for a whole day of driving, so the batteries have to be replaced or recharged during the day.

In this chapter, we present two models for scheduling electric vehicles in public transport. Aside from the classic constraints of vehicle scheduling, the models allow us to take the specific constraints for EVs into account. For simplicity, we will consider only the case of one depot and one vehicle type. Extending the Electric Vehicle Scheduling problem (E-VSP) to multi-depot and multiple vehicle types is similar to extending the traditional VSP to multi-depot and multiple vehicle types.

The two models that we present differ in the level of detail in resembling the actual electricity charging and usage process. In our first model, we assume a charging process that is linear in time, work with a constant price of electricity during the day, and do not take the effect of the depth-of-discharge on the lifetime of the battery into account. Our second model resembles practice much better: we allow any type of charging process, work with the actual electricity prices, and take the depreciation cost of the battery into account. To keep this model tractable, however, we approximate the exact value of the charge by discretization.

[1] The major part of this chapter has been published in M.E. van Kooten Niekerk, J.M. van den Akker, and J.A. Hoogeveen. Scheduling electric vehicles. *Public Transport*, 9(1):155-176, Jul 2017. ISSN 1613-7159.

Furthermore, in order to make a fair comparison between the two models, we only test the linear charging process in both models.

The remainder of the chapter is organized as follows. In Section 5.2 we present an overview of the literature on the E-VSP. Next, we describe the differences between traditional and electric buses in Section 5.3, and in Section 5.4 we mention the properties of EVs that we take into account. In Section 5.5, we present two models and a number of solution methods for the E-VSP, which are tested and evaluated in Section 5.6. For our experiments, we use data provided by De Lijn, a public transport organization in Belgium. Finally, we present our conclusions and directions for future research in Section 5.7.

5.2 Literature overview

In this section, we discuss some of the relevant literature on the subject of electric vehicle scheduling in public transport.

As to the computational complexity of these vehicle scheduling problems, Lenstra and Rinnooy Kan [1981] show that the traditional VEHICLE SCHEDULING PROBLEM can be solved in $\mathcal{O}(n^3)$ time and that the MULTI-DEPOT VEHICLE SCHEDULING PROBLEM is NP-hard. For an appropriate formation of the E-VSP, NP-hardness is proven in Sassi and Oulamara [2017].

Both Adler [2014] and Li [2014] discuss the E-VSP, but they consider EVs with a replaceable battery. For solving the E-VSP, they use COLUMN GENERATION, where the master problem is selecting a set of vehicle tasks in order to drive all trips, and where the subproblem is finding a vehicle task in order to improve the current solution of the master problem. Our approach is similar to theirs. For solving the subproblem, they model the trips and possible links between them as a graph and look for a shortest path. For this, the authors used a pricing algorithm based on the RESTRICTED SHORTEST PATH PROBLEM, which is described in Section 2.8 and known to be NP-hard (Huang and Li [2016]). In Adler [2014], the author also discusses the Concurrent Scheduling heuristic, which solves the E-VSP very fast, but at the expense of a solution which is between 10% and 15% from the optimal solution.

In Reuer et al. [2015], the authors solve a version of the VEHICLE SCHEDULING PROBLEM with a fleet consisting of electric vehicles and diesel vehicles without range restrictions. They model this as a time-spaced network and allow *opportunity charging* at given locations, where a battery can be charged during a vehicle task between two trips outside the depot. The authors assume that the battery can be fully charged in 10 minutes. If it would take longer, they assume that the battery can be replaced in 10 minutes. They use six different ways of flow decomposition and a heuristic inspired by Adler [2014]. The authors show that in their instances, 18 to 100 percent of the diesel vehicles can be replaced by EVs, which they consider to be sufficient for the transition phase from a full diesel fleet to a full electric fleet.

A problem similar to the E-VSP is the Electric Vehicle Routing Problem with Time Windows and Recharging Stations (E-VRPTWRS), which is defined and discussed in Bruglieri et al. [2015] and Schneider et al. [2014]. When the trips in the E-VSP are modeled as customers with a time window of width zero in which only the exact departure time is allowed, we obtain an instance of the E-VRPTWRS, which could be solved by the heuristics in these

papers. In both papers, Variable Neighborhood Search is applied. In Bruglieri et al. [2015], this is combined with Local Branching and in Schneider et al. [2014] it is combined with Tabu Search and Simulated Annealing. The authors show that their methods are faster than other known methods, but the datasets they use are small in comparison to the datasets that are used in the public transport domain. In order to determine the applicability of their heuristics in public transport, more testing should be done. The reason is that the conclusion of their work is that the heuristic performs better for large time-windows, whereas the public transport application works with time windows of length 0.

Huang and Li [2016] describe a shortest path algorithm specifically designed for electric bus scheduling with battery replacement. They use a label-correcting algorithm to be able to keep track of the State of Charge (SoC) during finding the best solution. For every trip, multiple labels can be stored, each referring to a path from the depot at the start of the day which could eventually lead to the shortest path. The graph they use is roughly equal to ours in Model 1, described in Section 5.5.1. We apply a modified version of this method for solving the pricing problem of COLUMN GENERATION when solving the E-VSP, with an adjustment for battery charging instead of battery replacement and with the addition of dual cost to the arcs. The label-correcting algorithm described in Huang and Li [2016] has an exponential running time, whereas our adjusted method runs in linear time on an expanded graph.

As far as we know, there is no previous research on the E-VSP that takes the battery depreciation and the non-linearity over time in energy and charging into account.

5.3 Problem description

The goal of the E-VSP is to determine the optimal vehicle schedule, given a set of trips and taking all constraints regarding EVs into account. The constraints of the E-VSP start with the traditional constraints from the VEHICLE SCHEDULING PROBLEM. Given a set of trips with fixed departure and arrival places and times and fixed travel times between all places, determine a set of vehicle tasks (i.e. what a vehicle drives on one day) where:

- Every trip is assigned to exactly one vehicle task,
- Every vehicle task drives a feasible sequence of trips, and
- The overall costs are minimal.

For this chapter, we only discuss the single depot situation with a single vehicle type. Using more than one depot or vehicle type is a straightforward extension of the model, similar to modeling the regular multi depot vehicle scheduling problem, and our methods can be used to solve these extensions as well.

For the E-VSP, the main difference that we have to take into account is that an EV has a battery that contains a limited amount of energy that is typically not enough for a whole day of driving. So we have some additional constraints that should be observed:

- At all times, the amount of energy in the battery of an EV should be sufficient to drive to the next charging station or the depot.

- At a given set of locations, the battery can be charged. This takes time and must be done when the vehicle is standing still. It may also be possible to exchange the empty battery with a full one. For this chapter, we do not allow battery exchange.

For the traditional VEHICLE SCHEDULING PROBLEM, the objective function reflects the cost of a solution, which is the sum of the fixed cost per vehicle needed and the variable cost per kilometer or minute for fuel, maintenance and crew. For the E-VSP, we have the same cost, but the objective function for the E-VSP will also contain the cost of battery depreciation, because the battery has a limited lifespan that is typically much shorter than the lifespan of the vehicle.

In the following section, we discuss the characteristics of the practical situations that are relevant to the E-VSP.

5.4 Properties of Electric Vehicles

In this section we will describe some knowledge that is needed in the problem of scheduling electric vehicles (EVs) in public transport.

5.4.1 Energy cost

The electricity needed for charging an electric vehicle comes from the electricity grid. As described on website `http://mpoweruk.com` by Lawson [2014], the consumption of electricity in the complete grid varies over the day; usually there is a peak consumption around the end of the afternoon. The level of the peak determines the capacity needed for the power grid and the power plants, and therefore we see at various electricity companies a Time-of-Use pricing in order to encourage consumers to use electricity outside the peak hours. Because the price of electricity may vary significantly over the day, we want to include this in our model. In our model, we will not assign the cost to the time when the electricity is consumed, but to the time when the electricity is taken from the grid and is put in the battery, because this time determines the electricity cost.

5.4.2 Charging infrastructure

Aside from electricity, we also need facilities to charge the vehicles. Such a charging station has a connection to the electricity grid and has equipment to transfer the electricity to the vehicle, for example a power cable or an induction loop. Charging stations can be built at any location, as long as there is a connection to the electricity grid and enough space where vehicles can charge. The most likely places are depots and terminals of routes.

Every charging station has associated properties and costs:

- Location. The construction cost of a charging station may vary due to ground prices, cooperation of the authorities, and availability of a high-power electricity connection in the vicinity.

- Charging capacity (space). For every location, a maximum number of vehicles can be charged simultaneously. This number depends on the space available.
- Charging capacity (energy). The capacity of the electricity connection may vary per charging station. With a larger capacity, EVs may be charged faster or more EVs can be charged simultaneously. However, this requires a larger power cable and will be more expensive.

For our problem formulations, we assume that the location of the charging stations and the properties of the stations are known. Optimization of the charging structure is outside the scope of this chapter.

5.4.3 Battery properties

An elaborate description of battery properties can be found in Lawson [2014] and Buchmann [2014]. The most important properties will be described in the following sections.

5.4.3.1 Battery capacity

A battery is manufactured with a given capacity. This is the amount of energy that can be stored at standard operating conditions when the battery is new. When the temperature is low or very high, the capacity can be reduced drastically, by tens of percents. This can be prevented by heating or cooling the battery. When this is not done, we have to take the reduced capacity into account by making different schedules for different available capacities. For example: we create one schedule for the summer and one schedule for the winter, when the battery capacities are 70% of the usual capacity.

5.4.3.2 Battery lifetime

During the lifetime of a battery, the capacity reduces because of chemical processes that occur inside the battery due to the usage of the battery. The lifetime of a battery is usually specified in *Cycle Lifetime*, which is the number of times that the battery can be fully discharged until it is considered end-of-life, which is when, measured at room temperature, the capacity of the battery is 80% of the original capacity when the battery was new. The actual lifetime of a battery is not determined by the number of charge/discharge cycles, but by the amount of energy that has been stored in total. Charging and discharging the battery for 10% of its capacity can be done ten times the number specified as Cycle Lifetime until the battery is end-of-life.

A second important factor is the Depth of Discharge (DoD). Discharging a battery fully will dramatically reduce its lifetime due to chemical processes that occur in the battery. When we have a battery with a Cycle Lifetime of 1000, then discharging this battery for 10% can be done 10000 times. In practice, the number will be higher. In Figure 5.1, a graph is shown for a Li-ion battery showing the number of recharge cycles related to the DoD. For every amount of energy in the battery (state of charge, SoC), we can calculate the cost per energy unit. When we want to use this in our models, we have to know what the SoC was before and

after charging and use the average cost per kWh in this range. The next paragraphs show an example of a calculation. For other batteries, the calculation will be similar.

We start with finding a formula that calculates the number of charge/recharge cycles $cycles(x)$ until end-of-life of the battery, starting with a fully charged battery and discharging up to a DoD x where $x \in [0, 1]$. For this, we use the numbers from Figure 5.1 and fit a function to it using the minimum least squares method. Evaluating miscellaneous function families, we get the best fit for an exponential function. The best fit found for the values shown in the graph is:

$$cycles(x) = 4825.3e^{-2.519x} \tag{5.1}$$

When we use $cost_{battery}$ to denote the cost of buying a battery, the cost $cost(x)$ of one cycle between an fully charged battery and a DoD x is assumed to be equal to

$$cost(x) = \frac{cost_{battery}}{cycles(x)} = \frac{e^{2.519x}}{4825.4} cost_{battery} \tag{5.2}$$

The cost $z(x_1, x_2)$ of one cycle of charge/discharge between a DoD of x_1 and a DoD of x_2, where $x_1 \geq x_2$ is approximated by

$$z(x_1, x_2) = cost(x_1) - cost(x_2) = \frac{e^{2.519x_1} - e^{2.519x_2}}{4825.4} cost_{battery} \tag{5.3}$$

Note that this formula calculates only the cost related to battery depreciation of a charge/discharge cycle. The cost of the energy itself is not included in this formula.

5.4.3.3 Battery charging characteristics

Charging a battery is not as simple as it may look. For every kind of battery, the chemistry of the battery should be taken into account. A charger for one type of battery is usually not capable of charging another type. For the Li-ion battery (which is the kind of battery that is used for most electric vehicles), a charger has to follow a complex charging scheme. This charging scheme implies that until 80% of full charge, the battery is charged quickly and after that, charging will be slowed down in order not to overheat the battery. In practice, charging a battery from 0% to 80% is a linear process, and it will take about the same time as charging the battery from 80% to 100%, where charging gradually slows down. In Section 5.5.2, we incorporate this in the graph of Model 2 by creating arcs between possible states of charge.

5.5 Models to solve the E-VSP

In order to solve the E-VSP, we model it in two different ways. In Model 1, we use a standard ILP-model for the VSP, to which we add continuous variables to track the charge of the batteries. In Model 2, we extend the underlying graph of the VSP-model in order to keep track of the charge. Every node that represents a trip is replaced by a set of nodes, where every node represents a possible state of charge on the trip. We use three different approaches to solve all models, which we refer to as Models 1a, 1b, 1c, 2a, 2b and 2c.

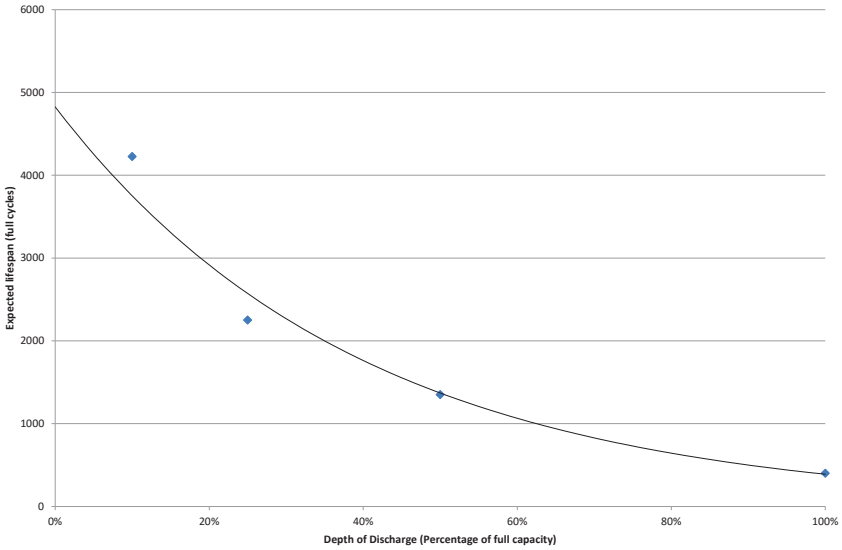


Figure 5.1: Graph showing the relation between DoD and the lifetime of a Li-ion battery, measured in number of recharge cycles. This is measured by repeatedly discharging the battery to the DoD on the horizontal axis. On the vertical axis, the number of charge/recharge cycles until end-of-life is indicated. Graph based on data from page BU808 of Buchmann [2014]

In Models 1a and 2a, described in Sections 5.5.1 and 5.5.2, we formulate the problem as an ILP and solve it with IBM ILOG CPLEX 12.2. In Models 1b and 2b, described in Section 5.5.3, we use COLUMN GENERATION in order to get a good solution and in Model 1c and 2c, we use COLUMN GENERATION in combination with LAGRANGIAN RELAXATION, which also gives a good, but not necessarily optimal solution for the E-VSP. For the pricing problem of COLUMN GENERATION in Models 1b and 1c we use the method described in Huang and Li [2016], for Models 2b and 2c we use a method especially designed in this thesis.

Each one of the models we distinguished has different properties. These properties are shown in Table 5.1.

Property	Model 1a E-VSP continuous	Model 1b and 1c cont. w. CG	Model 2a E-VSP discrete	Model 2b and 2c discr. w.CG
Charge variable	Exact	Exact	Rounded	Rounded
Time-of-Day pricing electricity	No	Yes	Yes	Yes
Non-linearity of charging time	No	Yes	Yes	Yes
Effects DoD on lifetime	No	Yes	Yes	Yes
Maximum problem size	Small/Medium	Large	Small/Medium	Large
Optimal solution guaranteed	Yes	No	Yes	No

Table 5.1: Properties of E-VSP models

5.5.1 Model 1a: E-VSP with continuous variables for battery charge

In our first model, we formulate the E-VSP by an ILP in the same way as the VEHICLE SCHEDULING PROBLEM: one node per trip and one node at the depot for every possible arrival or departure time. The nodes at the depot are used to keep track of the number of vehicles parked at the depot. For every combination of two trip nodes or a trip node and a depot node, we create an arc when these two can be assigned to the same vehicle. This arc is a connection between two trips and can contain deadhead trips, charging at a charging station and standing still, waiting for the next trip. All cost involved with this are associated with the arc, including the cost of the electricity that is used during the trip and the deadhead trip that may be associated with this arc.

For every trip, we assign an extra variable that keeps track of the charge at the start of a trip. For every node and arc we calculate the difference in charge and use this in the model. Ignoring the battery properties as described in Section 5.4.3, we assume that charging is a linear process and that battery depreciation is linear to the amount of energy used. As we can see in Buchmann [2014], the behavior of the battery is linear when the battery is charged for 80 percent or less, so when we use only 80 percent of the capacity in our model, this assumption is valid. This assumption of linearity is necessary because we want to model the E-VSP as an INTEGER LINEAR PROGRAMMING.

In the graph, we create for every trip i a node n_i. The collection of all nodes n_i is denoted by N. For every minute $t \in \{0, 1, ..., 1919\}$[2] on the depot we also create a node d_t. For every

[2]We continue after midnight until the next morning 8 am, because many timetables end after midnight. In this way, we are still able to model them. Note that $1920 = 32 \times 60$.

two trips i and j that may follow each other, we create an arc (n_i, n_j) with associated variable a_{ij}. At the depot, for every $t \in \{0, 1, ..., 1918\}$ we create arcs (d_t, d_{t+1}) with associated variable g_t representing the number of vehicles at the depot just after time t. These are single arcs that may represent more than one vehicle: g_t may be larger than 1. We further use g_{1919} to represent the total number of required vehicles, because all vehicles will be at the depot at the end of the day. We also create arcs that represent pull-in and pull-out trips to and from the depot. A pull-out arc (d_t, n_i) is represented by variable p_{ti}, a pull-in arc (n_i, d_t) is represented by variable q_{it}. In our ILP formulation, the variables a_{ij}, p_{ti} and q_{it} are binary variables, and g_t is an integer variable that represents the number of vehicles in the depot just after time t.

This leads to the following constraints. For every node n_i we require that there is exactly one incoming and one outgoing arc:

$$\sum_{j:n_j \in N} a_{ji} + \sum_{t=0}^{1919} p_{ti} = 1 \text{ for all } i \tag{5.4}$$

$$\sum_{j:n_j \in N} a_{ij} + \sum_{t=0}^{1919} q_{it} = 1 \text{ for all } i \tag{5.5}$$

For every node d_t with $t \in \{1, ..., 1919\}$ at the depot, we make sure that g_t is equal to the number of vehicles at the depot:

$$\sum_{i:n_i \in N} q_{it} + g_{t-1} = \sum_{j:n_j \in N} p_{tj} + g_t \text{ for all } t \tag{5.6}$$

Furthermore, we need to add constraints to enforce that for every depot, the number of vehicles at the start of the day is equal to that at the end of the day:

$$\sum_{i:n_i \in N} q_{i1919} + g_{1919} = \sum_{j:n_j \in N} p_{0j} + g_0 \tag{5.7}$$

For this case, where we consider only one depot, the latter equation is obsolete.

Until here, these were the constraints for a standard VEHICLE SCHEDULING PROBLEM formulation. For tracking the charge of the battery, we use some additional variables and constraints.

For every trip i, we have a variable $x_i \in \mathbb{R}^+$, indicating the charge at the start of this trip, and we have a parameter $u_i \in \mathbb{R}^+$, for the usage of energy to drive this trip. For every arc a_{ij} , we define a parameter $v_{ij} \in \mathbb{R}^+$ for the consumption of energy when it is necessary to drive a deadhead trip, and a parameter $w_{ij} \in \mathbb{R}^+$ for the maximum amount of energy that can be charged on this arc. For now, we assume that charging takes place after a deadhead trip and only at the given charging station locations. For every arc (d_t, n_j) we define a parameter $\pi_{tj} \in \mathbb{R}^+$ and for every arc (n_i, d_t) we define a parameter $\rho_{it} \in \mathbb{R}^+$ denoting the energy consumption during this arc.

For every trip i, we require that there is enough energy in the battery to complete the trip:

$$x_i \geq u_i \tag{5.8}$$

For every trip i, the charge at the begin of the trip may not exceed the maximum charge $x_{max} \in \mathbb{R}^+$:

$$x_i \leq x_{max} \tag{5.9}$$

For every arc between two trips i and j that contains a deadhead trip, we make sure that there is enough energy at the start of trip i in order to drive trip i and the deadhead trip. For this, we extend Equation (5.8) with the energy consumption of the deadhead trip:

$$x_i \geq u_i + a_{ij}v_{ij} \tag{5.10}$$

Because every trip has an outgoing arc, we may omit Equation (5.8). We also calculate the charge at the start of trip j, where M is an arbitrary large number. For arcs from trip i to trip j we use:

$$x_i - a_{ij}u_i - a_{ij}v_{ij} + a_{ij}w_{ij} + (1 - a_{ij})M \geq x_j \tag{5.11}$$

Note that we only charge when the start of trip j is at a charging station. For arcs between the depot and a trip j and a maximum SoC of SoC_{max} we use:

$$SoC_{max} - p_{tj}\pi_{tj} \geq x_j \tag{5.12}$$

And for arcs between a trip i and the depot we use:

$$x_i \geq q_{it}(u_i + \rho_{it}) \tag{5.13}$$

The objective function of the model now consists of two components:

- Fixed cost per vehicle. This is put in the objective by multiplying the cost per vehicle with g_{1919}, assuming that there are no overnight trips.
- Variable cost per vehicle. This is calculated for every arc and contains the cost on duration, distance and charged energy of the arc itself and the following trip.

5.5.2 Model 2a: E-VSP with discrete variables for battery charge

The model from Section 5.5.1 does not allow the charging time to be non-linear, does not allow the variable cost to be dependent on the SoC, and does not allow Time-of-Day Pricing of energy. Because these factors may have a large impact on the cost, we develop a second model. Our second model is largely similar to Model 1 as described in the first paragraphs in Section 5.5.1, but with the difference that we do not keep track of the charge in one single continuous variable per trip. Furthermore, the cost for energy associated with an arc is not the energy used in the trip before and during the arc, but the energy that was charged during the arc. In this way, we can include the Time-of-Day Pricing in our model.

For every trip we create a set consisting of the nodes that represent the combination of trip and SoC of the battery at the start of a trip. To keep the size of the model tractable, we discretize the charge for every trip. Therefore these values are not exact, but now we are able to take most battery properties from Section 5.4.3 into account.

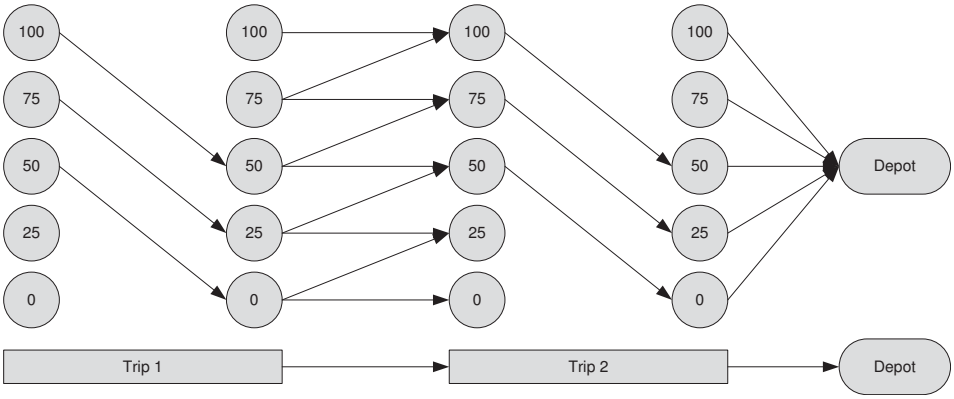


Figure 5.2: Graph representing two trips and one depot at the end. The circles are the nodes, representing the start or end of a trip in combination with the charge of the vehicle. Trips 1 and 2 both cost 50 units of energy and between trips 1 and 2 it is possible to charge either 0 or 25 units of energy.

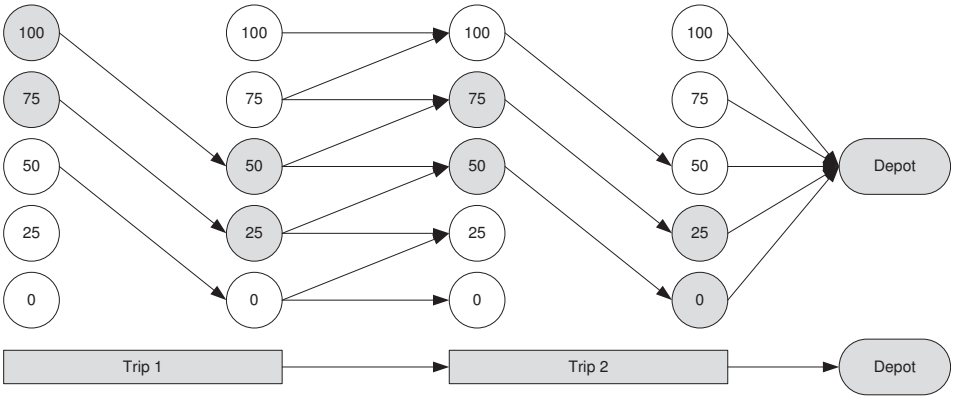


Figure 5.3: The white nodes are unreachable, they can be omitted.

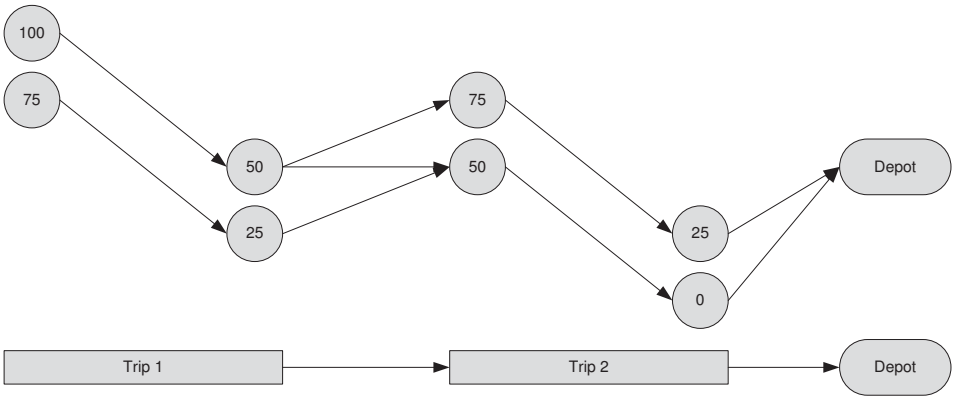


Figure 5.4: Unreachable nodes and unusable arcs are deleted.

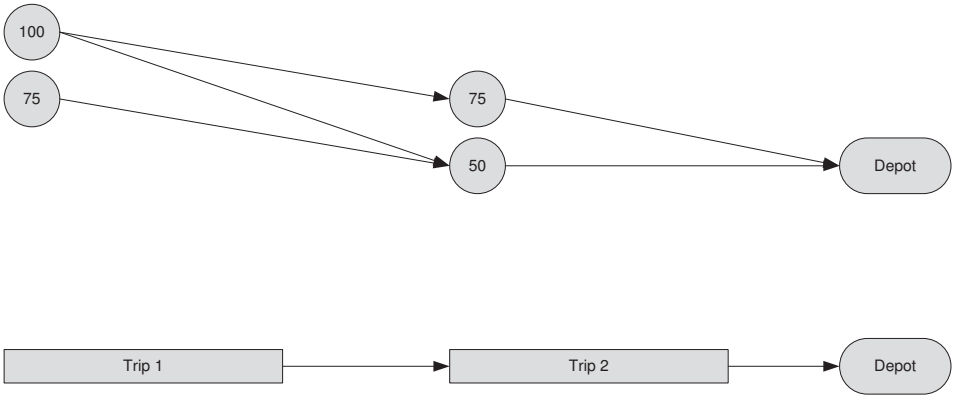


Figure 5.5: Trip end nodes are removed, because there is a one-to-one relationship between trip start nodes and trip end nodes.

In Figures 5.2 to 5.5, we show the construction and subsequent reduction of the graph used for this problem. We show a graph representing two trips and a depot at the end of the block. In Figure 5.2, we have drawn nodes for every start and end of a trip in combination with the electric charge of the vehicle at that moment. Arcs represent the allowed sequence of nodes; if it is possible to charge in between these trips, then there is usually more than one arc per node, because it is possible to charge the vehicle at full power, not charge it at all or anything in between. In Figure 5.3, we have whitened the nodes that are unreachable, because they do not have any outgoing or incoming arc. In Figure 5.4, these nodes have been removed, together with the now obsolete arcs. We note that for every trip, there is a one-to-one relationship between a trip start node and a trip end node, so we combine them to the trip start nodes. This can be seen in Figure 5.5.

For every trip i with a SoC at its start of $x_i \in \mathbb{R}^+$, we create a node n_{ix} with associated variable $\eta_{ix} \in \{0,1\}$ denoting the use of the node. All these nodes together form the collection N. For the SoC value x, we want to use $steps + 1$ different values, so we only use values that are a multiple of $\frac{1}{steps}$. The collection of all possible values for the SoC is X. In Figures 5.2 to 5.5, we use $steps = 4$ and in our experiments in Section 5.6 we use $steps = 50$. When a combination of trip and charge is not possible, because the vehicle does not have enough charge to complete the trip, the node is not created. For every allowed combination of two nodes n_{ix} and n_{jy}, we create an arc (n_{ix}, n_{jy}) with associated variable a_{ixjy}. In this way, when there is enough time between trips to charge, we can also include the decision whether to charge or not on a charging station, as well as the amount of energy to charge.

For the depot, we also create a set of nodes. For every time unit t and SoC of $\sigma \in X$, we create node $d_{t\sigma}$. Between every two adjacent nodes $d_{t\sigma}$ and $d_{t+1,\sigma'}$, where σ' is the SoC when the battery is charged from a SoC of σ during one time unit, we create a depot occupation arc $(d_{t\sigma}, d_{t+1,\sigma'})$ with associated variable $g_{t\sigma}$, which represents the number of vehicles that is at the depot at time unit t with SoC σ, where $g_{t\sigma} \in \mathbb{N}^0$. We also create a depot occupation arc $(d_{1919\sigma}, d_{0\sigma})$ with variable $g_{1919\sigma}$. For deadhead trips from the depot $d_{t\sigma}$ to an in-service trip n_{ix} and vice versa we create arcs $(d_{t\sigma}, n_{ix})$ with variable $b_{out,t\sigma ix}$ for pull-out trips and $(n_{ix}, d_{t\sigma})$ with variable $b_{in,t\sigma ix}$ for pull-in trips. The variables $\eta_{ix} \in \{0,1\}$, $a_{ixjy} \in \{0,1\}$, $b_{out,t\sigma ix} \in \{0,1\}$ and $b_{in,t\sigma ix} \in \{0,1\}$ indicate the use of a node or an arc and are binary.

Using these variables, we can now formulate the problem as an INTEGER LINEAR PROGRAMMING problem with the following constraints. Every trip i should be covered by one vehicle:

$$\sum_{x \in X} \eta_{ix} = 1 \text{ for all } i \tag{5.14}$$

There should be exactly one arc to every in-service trip i, if it is used. Note that there is at most one arc $b_{out,t\sigma ix}$ and $b_{out,t\sigma' ix}$ for every combination of i and x, since the remaining charge x follows immediately from σ.

$$\sum_{j,y:n_{jy} \in N} a_{jyix} + \sum_{t=0}^{1919} b_{out,t\sigma ix} = \eta_{ix} \text{ for all } i, x \tag{5.15}$$

And there should be exactly one arc from every in-service trip i, if it is used:

$$\sum_{j,y:n_{jy}\in N} a_{ixjy} + \sum_{t=0}^{1919} b_{in,t\sigma ix} = \eta_{ix} \text{ for all } i, x \tag{5.16}$$

For the depot nodes and arcs, the formulas are somewhat different. We assume that nodes $d_{t\sigma}$ exist for $t \in \{0, 1, ..., 1919\}$ and all σ, being from midnight today to 8 o'clock in the morning the next day. We require the number of incoming vehicles at every depot node to be equal to the number of outgoing vehicles:

$$\sum_{i,x:n_{ix}\in N} b_{in,t\sigma ix} + g_{t\sigma} = \sum_{i,x:n_{ix}\in N} b_{out,t\sigma' ix} + g_{t+1,\sigma'} \text{ for all } t < 1919, \sigma \tag{5.17}$$

Then we make sure that the number of vehicles at the start of the day is equal to the number at the end of the day.

$$\sum_{i,x:n_{ix}\in N} b_{in,1919\sigma ix} + g_{1919\sigma} = \sum_{i,x:n_{ix}\in N} b_{out,1919\sigma ix} + g_{0\sigma} \text{ for all } \sigma \tag{5.18}$$

5.5.3 Models 1b and 2b: COLUMN GENERATION

Later tests show that both Models 1a and 2a in Sections 5.5.1 and 5.5.2 give good solutions for the E-VSP. However, in practice we see that both methods are too slow to be applicable in real-life instances with more than 10 vehicles. In this case, we may use the technique of COLUMN GENERATION to find a good, although not necessarily optimal solution (see for example Desaulniers et al. [2005]).

For this, we redefine the INTEGER LINEAR PROGRAMMING problem we use. In the new Master Problem (MP), we use variables to represent every possible vehicle task, and we use constraints to impose that every trip is planned in at least one vehicle task. Since the number of possible vehicle task is huge, we will use a restricted version of this MP, which contains a selection of vehicle tasks. Using a process called COLUMN GENERATION, we add new vehicle tasks to the Restricted Master Problem (RMP) in order to improve the cost.

For the definition of the RMP, we look for the optimal set of vehicle tasks that covers all trips, where a vehicle task is a set of trips that can be driven by one vehicle on a day, taking all constraints with respect to the charge of the EV into account. Let K be the collection of all valid vehicle tasks in the RMP. We use variables $b_k \in \{0, 1\}$ for every valid vehicle task $k \in K$ to indicate if that vehicle task is used in the solution; we use c_k to denote the cost of vehicle task $k \in K$. Furthermore, we use binary parameters $r_{ki} \in \{0, 1\}$ to indicate if trip i is part of vehicle task k.

For the objective function, we use:

$$\text{Minimize: } \sum_{k\in K} c_k b_k \tag{5.19}$$

For every trip i, we require that it is part of at least one vehicle task:

$$\sum_{k\in K} r_{ki} b_k \geq 1 \text{ for all } i \tag{5.20}$$

$$b_k \in \{0, 1\} \text{ for all } k \tag{5.21}$$

When we get a result for Equation 5.20 that is larger than 1, this will lead to multiple vehicles on one trip. In this case, we choose one vehicle to drive the trip and the other vehicle will drive the trip as a deadhead trip.

At the start, we fill the RMP with dummy vehicle tasks, one for every trip. These dummy vehicle tasks each contain one trip; when we run the COLUMN GENERATION, these columns will be discarded soon.

We start with relaxing the integrality constraints: instead of the constraint $b_k \in \{0, 1\}$, we require $b_k \geq 0$. When solving this LP relaxation of this RMP, we get a dual cost for every constraint, that is for every trip. In order to improve the solution of the RMP, we look for extra columns (vehicle tasks) with negative reduced cost. The reduced cost of a column is the cost c_k of the column minus the sum of the dual cost of all trips included in the vehicle task.

The pricing problem is to minimize reduced cost. If it finds vehicle tasks with negative reduced costs, these are added to the RMP. If not, the optimal solution of the RMP was found. Hereto, we use the graphs from Sections 5.5.1 and 5.5.2 and for every arc that ends in a trip, we subtract the dual cost of that trip. Then we look for the path with minimum cost from d_0 to d_{1919}. For Model 1b, we have to keep track of the SoC of the vehicle, while in Model 2b this is incorporated in the graph. So for Model 1b we have to use an adapted version of the shortest path algorithm to take this into account. We use a label-correcting algorithm, which is proposed for this use by Huang and Li [2016] and is briefly described in Section 2.8. Note that in this label-correcting algorithm, the cost of an arc is only known after the properties of the preceding node are known, including the SoC, so we can make the cost of an arc dependent on the SoC at the end of the previous trip. In this way, we implement the battery wear and tear and the non-linear charging process from Section 5.4. Huang and Li [2016] show that the pricing problem above is NP-hard.

For Model 2b we just have to find the shortest path in the graph described in Section 5.5.2. This shortest path can be found in $\mathcal{O}(E)$ time, where E is the number of arcs, due to the fact that the graph is directed acyclic graph. Since the number of arcs is at most the number of possible combinations of nodes, which in turn is the number of trips times the given number of possible SoC values $steps + 1$, the number of arcs is $\mathcal{O}(T^2)$, where T represents the number of trips. Finding the shortest path can be done in $\mathcal{O}(T^2)$ time, so solving the pricing problem in Model 2b is polynomial in time for a given value $steps + 1$.

If the cost of a shortest path is non-negative, we cannot improve on the solution of the RMP, otherwise we add the column to the RMP and solve it again. We continue until the RMP has been solved to optimality.

At this point, we have a set of variables b_k, which indicate which vehicle tasks are part of the optimal solution. However, we do not have a guarantee that the values of these variables are integral. When we have a fractional result, we use several strategies to end up with an integer solution. We do this by reducing the size of the master problem, making heuristic decisions about arcs, nodes and vehicle tasks. The heuristics that we apply are:

- Inspired by Desrosiers et al. [2014], we analyze the result from Column Generation. For the columns that have a strictly positive result in the RMP, we look for identical rows. This means that the trips belonging to these rows always occur together in a

vehicle task. When these occur and these trips are consecutive, we merge these two trips and update the graph.

- If there are columns that have a result in the RMP larger than 0.95, then the value of these columns is set to 1, these columns are removed from the RMP, and the nodes and arcs that belong to these columns are removed from the graph. Then we solve the RMP again with the same columns, generating new columns if necessary. This is similar to the Truncated COLUMN GENERATION approach as described in Pepin et al. [2009].

- If there is any arc that is used by columns or vehicle tasks with a total value for b_k in the RMP of at least 0.99, we fix this arc, remove the columns that do not respect this fixed arc, and continue solving the RMP with the remaining columns.

- If there is any arc that is not used by any column or vehicle tasks or is used by columns and vehicle tasks with a total value for b_k in the RMP of less than 0.01, we remove this arc from the graph, all columns that use this arc from the RMP, and solve the RMP again with the same columns.

- If none of the above leads to a reduction of the graph and the RMP, we fix all columns with a value of 0.7 or more by setting their value to 1. If there are no such columns, we fix the column with the largest value. Then we solve the RMP again with the same columns and continue generating columns.

Since we need integral values for b_k, we repeat these steps until the RMP has an integral solution.

5.5.4 Models 1c and 2c: COLUMN GENERATION in combination with LAGRANGIAN RELAXATION

In Section 5.5.3, we used the duals from the LP relaxation for generating additional columns. In Huisman et al. [2005], an interesting alternative approach is explained to obtain dual values for use in COLUMN GENERATION. In this section, we use this alternative approach.

We start with an explanation of this alternative approach. In Equations 5.19 and 5.20, we have defined the RMP for COLUMN GENERATION:

$$\min \sum_{k \in K} c_k b_k \tag{5.22}$$

$$\text{s.t.} \sum_{k \in K} r_{ki} b_k \geq 1 \text{ for all } i \tag{5.23}$$

$$b_k \in \{0, 1\} \tag{5.24}$$

We introduce a *Lagrangian multiplier* $\lambda_i \geq 0$ with $\lambda_i \in \mathbb{R}$ for every constraint i, and put

the constraints in the objective function as follows:

$$\Phi(\lambda) = \min \sum_{k \in K} c_k b_k + \sum_i \lambda_i (1 - \sum_{k \in K} r_{ki} b_k) \quad (5.25)$$

$$\text{s.t.} \quad b_k \in \{0, 1\} \quad (5.26)$$

For every non-negative vector of Lagrangian multipliers λ, $\Phi(\lambda)$ gives a lower bound on the solution of the original ILP in Equations (5.22) to (5.24). Since we can rewrite the objective function to $\sum_{k \in K} b_k (c_k - \sum_{i:n_i \in N} \lambda_i r_{ki}) + \sum_{i:n_i \in N} \lambda_i$, we observe that the value for $\Phi(\lambda)$ in Equation (5.25) is minimized when we set $b_k = 1$ when $c_k - \sum_{i:n_i \in N} \lambda_i r_{ki} < 0$ and $b_k = 0$ otherwise. We now solve the Lagrangian Dual Problem, which is maximizing $\Phi(\lambda)$ while $\lambda \geq 0$. For this optimization, we use *subgradient optimization*.

The values for λ when $\Phi(\lambda)$ is maximal can be used as dual costs for the COLUMN GENERATION with the pricing problem described in Section 5.5.3. Unlike when the dual costs are determined by solving a RMP through LP, we cannot guarantee now that the already generated columns have nonnegative reduced cost. As a result, we may find an already known column when solving the pricing problem, which brings us no further. In order to prevent this, we apply the heuristic from Freling [1997] and Carraresi et al. [1995] that changes the vector λ so that all current columns will have a non-negative reduced cost and the value of the Lagrangian function $\Phi(\lambda)$ will not decrease. The heuristic is described in Algorithm 4. For a detailed description we refer to Freling [1997] and Carraresi et al. [1995].

```
1 foreach column k with c_k − Σ_{i:n_i∈N} λ_i r_ki < 0 do
2 |   δ ← (c_k − Σ_{i:n_i∈N} λ_i r_ki) / (Σ_{i:n_i∈N} r_ki) ;
3 |   foreach i with r_ki=1 do
4 |   |   λ_i ← λ_i + δ;
5 |   end
6 end
```

Algorithm 4: Heuristic to modify Lagrangian multipliers

After applying this heuristic to change λ, we use these multipliers as duals instead of the values that result from solving the RMP with a LP-solver as described in Section 5.5.3.

5.6 Computational results

To test our algorithms, we use data from the city of Leuven, provided by De Lijn. We will evaluate the cost of exploiting the urban routes 2, 3 and 600/601 with electric vehicles. A fixed cost per vehicle is taken into account, as well as a variable cost per kilometer for driver cost and electricity/fuel cost. For the given charging points, we assume that there is no limit on the number of simultaneous chargings taking place. We also assume that in case of a deadhead between two trips, the deadhead immediately follows the trip, after which a charging may take place. In the situation where these routes are all driven by buses running on fossil fuel,

27 buses are needed, which drive 543 trips of in total 337:51 hours per day. This schedule is shown in Figure 5.6.

For our optimizations, we use electric vehicles with two different capacities. Every optimization uses one vehicle type, so every optimization is done for every capacity. The characteristics of the electric vehicles are:

- Battery capacity: 122 or 244 kWh
- Energy usage: 1.2 kWh per km
- Charge speed: 2.0 kWh per minute

For Model 2 we use 51 nodes per trip to reflect the SoC in steps of 0.02. In order to compare Model 1 with these models, we use a linear charging duration instead of a more realistic non-linear charging duration.

For our calculations, we assume that charging can only take place at four specific sites. When the bus stops at one of those places, the battery will charge. The charging stations are located at

- Kessel-Lo, Hulsberg (Terminus of route 2)
- Leuven, Gasthuisberg Campus (Terminus of route 3)
- Leuven, Stelplaats Diestsepoort (Depot)
- Leuven, Vaartkom (Terminus of route 600/601)

In order to evaluate our algorithms, we have split the vehicle schedule in three parts, one part with all trips on route 2, one part with the trips on route 3 and one part with all trips on routes 600 and 601. For all three datasets, we executed all models for both battery capacities. Furthermore, to test the scalability, we also run Models 1b, 1c, 2b and 2c on a dataset with all four lines 2, 3, 600 and 601 together, which is called 'All Routes'. The results of these optimizations can be found in Tables 5.2 and 5.3. For comparison, we have also added the same data for traditional vehicle scheduling in this table. We have added illustrations of the solutions for traditional vehicle scheduling; these can be found in Figure 5.6 and for Model 2c in Figures 5.9 and 5.10.

All optimizations are run on a computer with an Intel®Core™i7-3770 microprocessor running on 3.4 GHz, with 16 GB RAM memory. We implemented the algorithms in Java 8, using IBM ILOG CPLEX 12.2 for solving LPs and ILPs.

When we compare the graphs of the traditional vehicle schedule in Figure 5.6 with the one of Model 2c and 244 kWh in Figure 5.10, we see that most of the schedule is the same. In the early morning and the early evening, extra layover is added to the schedule in order to charge the vehicles. The capacity of 244 kWh seems to be enough for vehicles that do not drive during the evening, but it is not enough for vehicles driving all day. For those vehicles, extra charging time needs to be scheduled.

Comparing the traditional vehicles schedule with the one of Model 2c and 122 kWh in Figure 5.9 shows that virtually every vehicle will run out of electricity if the traditional vehicle

			Battery Capacity 122 kWh					
Method	**Dataset**	**Trips**	**Veh.**	**In-Serv hrs**	**Dhd hrs**	**Idle hrs**	**TOTAL hrs**	**Dur. of Opt. (s)**
Trad	Route 2	241	12	159:30	4:32	16:08	180:10	0
Trad	Route 3	127	8	93:58	4:04	16:45	114:47	0
Trad	Route 600/601	175	7	84:23	1:04	7:02	92:29	0
Trad	**SUM**	**543**	**27**	**337:51**	**9:40**	**39:55**	**387:26**	**0**
1a - MIP	Route 2	241	13	159:30	5:08	26:16	190:54	464
1a - MIP	Route 3	127	8	93:58	4:20	21:00	119:18	104
1a - MIP	Route 600/601	175	7	84:23	1:04	10:43	96:10	326
1a - MIP	**SUM**	**543**	**29**	**337:51**	**10:32**	**57:59**	**406:22**	**894**
1b - CG	Route 2	241	13	159:30	5:16	28:11	192:57	10
1b - CG	Route 3	127	9	93:58	5:21	25:05	124:24	4
1b - CG	Route 600/601	175	8	84:23	1:36	11:15	97:14	5
1b - CG	**SUM**	**543**	**30**	**337:51**	**12:13**	**64:31**	**414:35**	**19**
1b - CG	All Routes	543	31	337:51	28:58	57:40	424:29	84
1c - CG with LR	Route 2	241	13	159:30	5:20	25:59	190:49	71
1c - CG with LR	Route 3	127	8	93:58	4:36	21:30	120:04	9
1c - CG with LR	Route 600/601	175	7	84:23	1:12	10:18	95:53	11
1c - CG with LR	**SUM**	**543**	**28**	**337:51**	**11:08**	**57:47**	**406:46**	**91**
1c - CG with LR	All Routes	543	29	337:51	26:15	58:01	422:07	816
2a - MIP with discr	Route 2	241	13	159:30	4:44	28:46	193:00	929
2a - MIP with discr	Route 3	127	8	93:58	4:20	23:04	121:22	19
2a - MIP with discr	Route 600/601	175	7	84:23	1:04	12:33	98:00	42
2a - MIP with discr	**SUM**	**543**	**28**	**337:51**	**10:08**	**64:23**	**412:22**	**990**
2b - CG	Route 2	241	13	159:30	4:56	28:39	193:05	1045
2b - CG	Route 3	127	8	93:58	4:20	23:48	122:06	141
2b - CG	Route 600/601	175	7	84:23	1:12	13:42	99:17	193
2b - CG	**SUM**	**543**	**28**	**337:51**	**10:28**	**66:09**	**414:28**	**1379**
2b - CG	All Routes	543	28	337:51	14:24	64:18	416:33	49330
2c - CG with LR	Route 2	241	13	159:30	4:56	29:21	193:47	102
2c - CG with LR	Route 3	127	9	93:58	5:31	22:52	122:21	27
2c - CG with LR	Route 600/601	175	7	84:23	1:04	13:10	98:37	32
2c - CG with LR	**SUM**	**543**	**29**	**337:51**	**11:31**	**65:23**	**414:45**	**161**
2c - CG with LR	All Routes	543	28	337:51	17:42	63:49	419:22	974

Table 5.2: Results for E-VSP for a vehicle with 122 kWh battery capacity

			Battery Capacity 244 kWh					
Method	**Dataset**	**Trips**	**Veh.**	**In-Serv hrs**	**Dhd hrs**	**Idle hrs**	**TOTAL hrs**	**Dur. of Opt. (s)**
Trad	Route 2	241	12	159:30	4:32	16:08	180:10	0
Trad	Route 3	127	8	93:58	4:04	16:45	114:47	0
Trad	Route 600/601	175	7	84:23	1:04	7:02	92:29	0
Trad	**SUM**	**543**	**27**	**337:51**	**9:40**	**39:55**	**387:26**	**0**
1a - MIP	Route 2	241	12	159:30	4:32	21:19	185:21	4
1a - MIP	Route 3	127	8	93:58	4:04	19:45	117:47	0
1a - MIP	Route 600/601	175	7	84:23	1:04	9:42	95:09	0
1a - MIP	**SUM**	**543**	**27**	**337:51**	**9:40**	**50:46**	**398:17**	**4**
1b - CG	Route 2	241	13	159:30	4:56	18:44	183:10	11
1b - CG	Route 3	127	8	93:58	5:02	17:59	116:59	17
1b - CG	Route 600/601	175	7	84:23	1:04	8:24	93:51	4
1b - CG	**SUM**	**543**	**28**	**337:51**	**11:02**	**45:07**	**394:00**	**32**
1b - CG	All Routes	543	28	337:51	21:22	41:38	400:51	224
1c - CG with LR	Route 2	241	12	159:30	4:44	18:25	182:39	39
1c - CG with LR	Route 3	127	8	93:58	5:02	17:06	116:06	4
1c - CG with LR	Route 600/601	175	7	84:23	1:04	7:02	92:29	6
1c - CG with LR	**SUM**	**543**	**27**	**337:51**	**10:50**	**42:33**	**391:14**	**49**
1c - CG with LR	All Routes	543	27	337:51	18:50	43:19	400:00	211
2a - MIP with discr	Route 2	241	12	159:30	4:32	26:10	190:12	1325
2a - MIP with discr	Route 3	127	8	93:58	4:04	21:04	119:06	15
2a - MIP with discr	Route 600/601	175	7	84:23	1:04	11:37	97:04	38
2a - MIP with discr	**SUM**	**543**	**27**	**337:51**	**9:40**	**58:51**	**406:22**	**1378**
2b - CG	Route 2	241	13	159:30	5:28	20:38	185:36	979
2b - CG	Route 3	127	8	93:58	4:20	18:48	117:06	128
2b - CG	Route 600/601	175	7	84:23	1:04	9:33	95:00	141
2b - CG	**SUM**	**543**	**28**	**337:51**	**10:52**	**48:59**	**397:42**	**1248**
2b - CG	All Routes	543	28	337:51	16:54	45:31	400:16	49290
2c - CG with LR	Route 2	241	12	159:30	4:32	19:36	183:38	92
2c - CG with LR	Route 3	127	8	93:58	4:52	18:03	116:53	25
2c - CG with LR	Route 600/601	175	7	84:23	1:04	10:21	95:48	26
2c - CG with LR	**SUM**	**543**	**27**	**337:51**	**10:28**	**48:00**	**396:19**	**143**
2c - CG with LR	All Routes	543	27	337:51	18:12	47:43	403:46	539

Table 5.3: Results for E-VSP for a vehicle with 244 kWh battery capacity

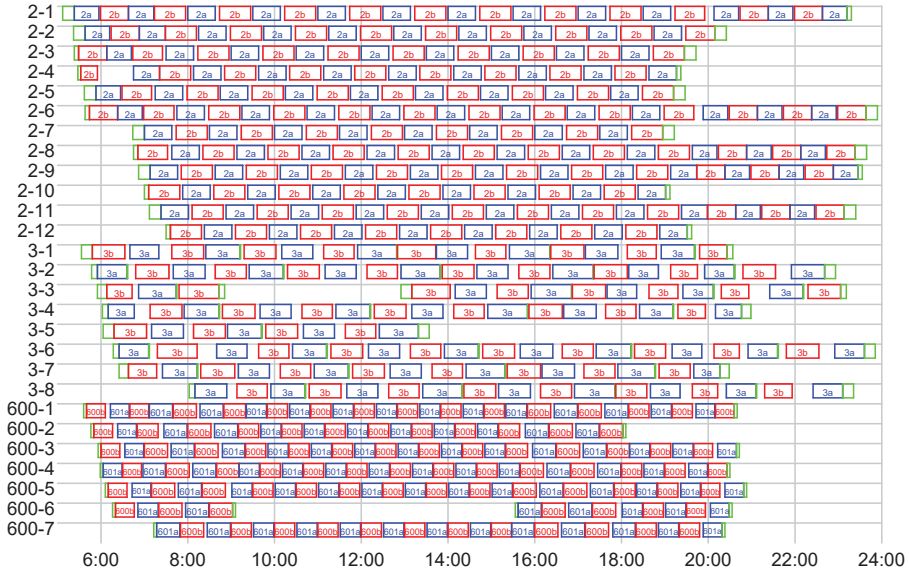


Figure 5.6: Vehicle schedule for the urban service of Leuven, based on the usage of fossil fuel powered vehicles. On the horizontal axis, the time of day is denoted. Every row shows the schedule of one vehicle. The red, blue and green rectangles represent the trips and the numbers inside a rectangle denote the route and direction of the trip.

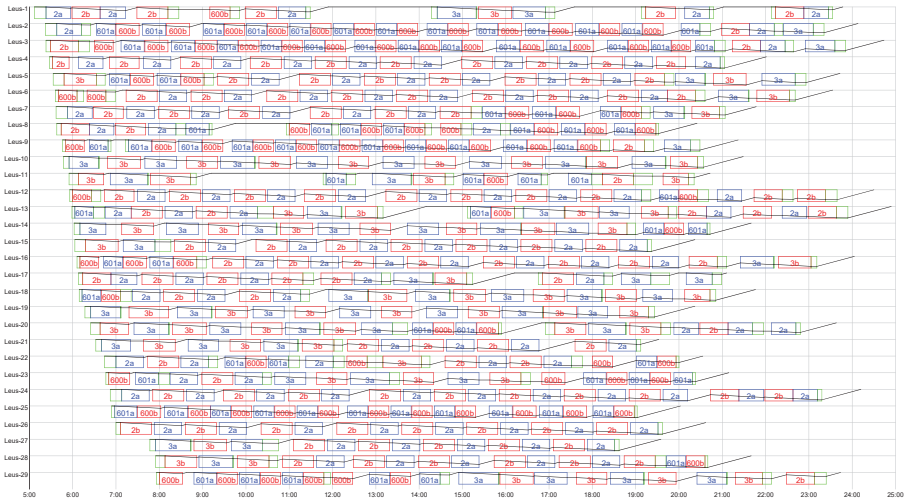


Figure 5.7: Vehicle schedule for the urban service of Leuven, using Model 1c (COLUMN GENERATION with LAGRANGIAN RELAXATION) for optimization and a battery capacity of 122 kWh. On the horizontal axis, the time of day is denoted. Every row shows the schedule of one vehicle. The red, blue and green rectangles represent the trips and the numbers inside a rectangle denote the route and direction of the trip. The black line through the trips shows the SoC during the day, where the top of the trip is a SoC of 1.0 and the bottom of the trip corresponds with a SoC of 0.0.

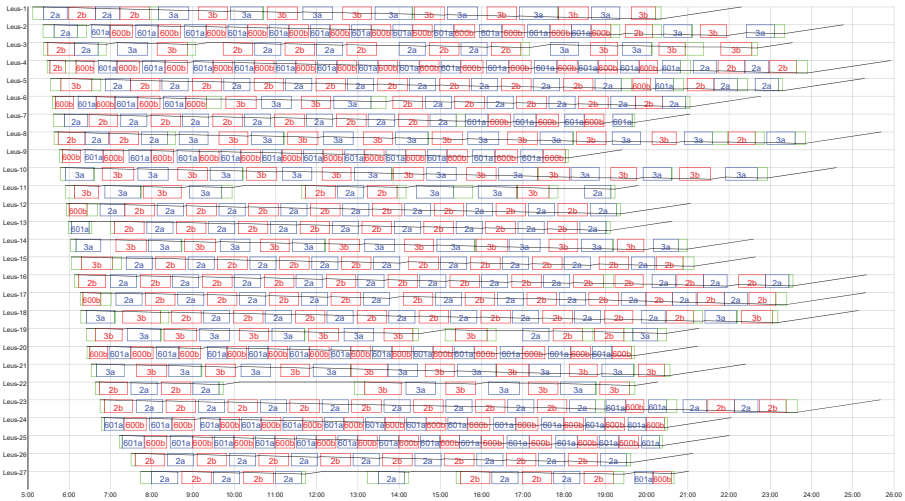


Figure 5.8: Vehicle schedule for the urban service of Leuven, using Model 1c (COLUMN GENERATION with LAGRANGIAN RELAXATION) for optimization and a battery capacity of 244 kWh. On the horizontal axis, the time of day is denoted. Every row shows the schedule of one vehicle. The red, blue and green rectangles represent the trips and the numbers inside a rectangle denote the route and direction of the trip. The black line through the trips shows the SoC during the day, where the top of the trip is a SoC of 1.0 and the bottom of the trip corresponds with a SoC of 0.0.

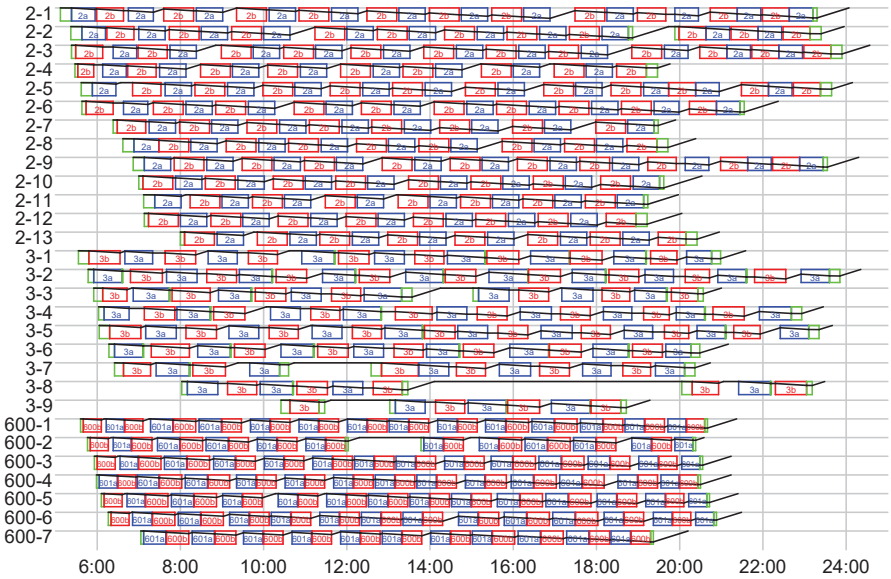


Figure 5.9: Vehicle schedule for the urban service of Leuven, using Model 2c (COLUMN GENERATION with LAGRANGIAN RELAXATION) for optimization and a battery capacity of 122 kWh. On the horizontal axis, the time of day is denoted. Every row shows the schedule of one vehicle. The red, blue and green rectangles represent the trips and the numbers inside a rectangle denote the route and direction of the trip. The black line through the trips shows the SoC during the day, where the top of the trip is a SoC of 1.0 and the bottom of the trip corresponds with a SoC of 0.0.

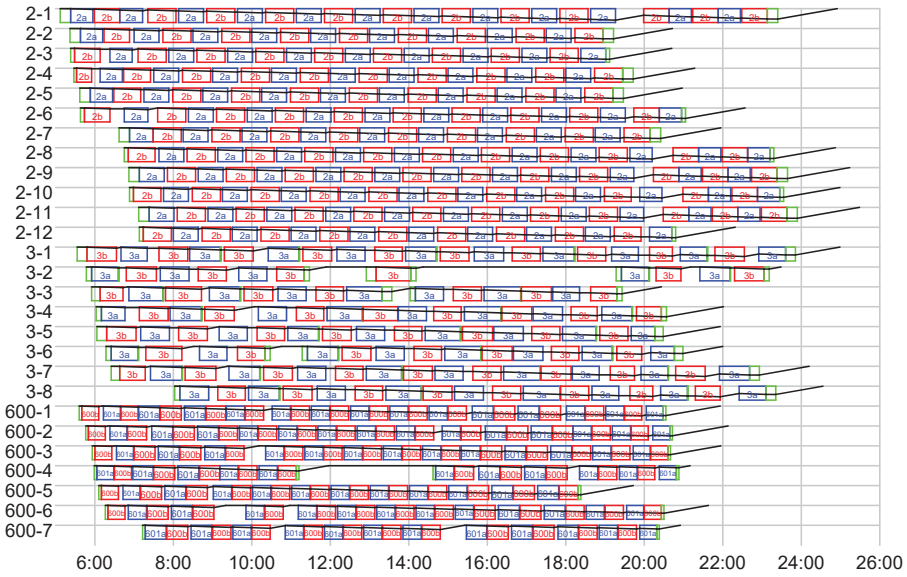


Figure 5.10: Vehicle schedule for the urban service of Leuven, using Model 2c (COLUMN GENERATION with LAGRANGIAN RELAXATION) for optimization and a battery capacity of 244 kWh. On the horizontal axis, the time of day is denoted. Every row shows the schedule of one vehicle. The red, blue and green rectangles represent the trips and the numbers inside a rectangle denote the route and direction of the trip. The black line through the trips shows the SoC during the day, where the top of the trip is a SoC of 1.0 and the bottom of the trip corresponds with a SoC of 0.0.

schedule is used. During the whole day, extra layovers are scheduled in order to charge the batteries.

Comparison of the models we proposed for solving the E-VSP shows that Model 1a gives the best results, followed by Models 2a, 2c, 1c, 2b and 1b. It was to be expected that Model 2a gives worse results than Model 1a, because Model 2a is an approximation of Model 1a. For the complex case of 122 kWh, we see in Table 5.2 that the total run times for Models 1a and 2a are comparable. For the easy case of 244 kWh however, we see in Table 5.3 that Model 1a is very fast. The disadvantage of Model 1a is that it can not handle non-linear charging schemes, but when this is not necessary, Model 1a is the best choice for instances with an almost sufficient battery capacity.

Models 1b, 1c, 2b and 2c give comparable results, which are only a few percent more expensive than Models 1a and 2a. The main difference between Models 1b, 1c, 2b and 2c is the runtime. Models 1c and 2c are much faster than Models 1b and 2b, and in Models 1c and 2c the problem with the dataset containing all routes is tractable in reasonable time, while it is not in Model 2b and it is not solved at all in Model 1b. Furthermore, Models 1b and 1c are faster for small instances, while Models 2b and 2c are much faster for large instances. For the 'All Routes'-datasets, we see that almost all results are more expensive than the results for the separate routes, which is the opposite of what we expected. We do not have an explanation for this, except that our method to acquire an integral solution as described in Section 5.5.3 is a heuristic that is not guaranteed to always give optimal results.

5.7 General conclusion

In this chapter, we have shown that the special properties of EVs must be taken into account in their scheduling, because we will end up with an infeasible vehicle schedule otherwise. For this purpose, we defined the E-VSP in Section 5.3 and have shown that in most cases, extra vehicles are needed for a feasible vehicle schedule.

For solving the E-VSP we have used two different models: Model 1 using a graph where every trip is represented by one node, and Model 2 where every combination of trip and State of Charge is represented by a node. On both models we used three different solution methods: INTEGER LINEAR PROGRAMMING (Models 1a and 2a), COLUMN GENERATION (Models 1b and 2b) and COLUMN GENERATION with LAGRANGIAN RELAXATION (Models 1c and 2c). For Models 2b and 2c, we showed that the pricing problem used in COLUMN GENERATION can be solved in polynomial time, whereas using the formulation of the problem by Huang and Li [2016] for Model 1 is NP-hard.

We have implemented the proposed two models and six solution methods in total to solve the E-VSP and tested them on four datasets. Comparing Model 1a to Model 2a shows us that both give us a solution with the same number of EVs, but that the solution of Model 2a needs more waiting time than Model 1a. This waiting time is used for the larger part for charging the vehicles. Because Model 2a is an approximation of Model 1a, it was to be expected that the resulting solutions were more expensive than in Model 1a. Models 1b, 1c, 2b and 2c give comparable results, where Model 1c is faster for small datasets and Model 2c is faster than all the other models in large datasets. Because Model 2c is capable to optimize the full dataset,

since it is faster, and since its final solution is not much more expensive than that of the other models, our experiments suggest that Model 2c is our first choice for solving the E-VSP.

We have tested the models on datasets with up to 30 vehicles. In real life, datasets could be much larger. Using the models for larger datasets could lead to less efficient results, an issue that could be subject of further investigation. Future work may be done on integrating the properties of charging infrastructure in the model, as described in Section 5.4.2. More research can also be done on allowing more possibilities for charging place and time between two trips; now it is assumed that charging always takes place after a deadhead, but it may be beneficial to charge before the deadhead or even at an intermediary place.

Chapter 6

Modeling the transition towards sustainable public transport

6.1 Introduction

Air pollution is a growing concern everywhere. On a global level, measures are taken in many countries to reduce pollution drastically. In the Paris Agreement (UNFCCC [2015]), almost all nations in the world have agreed on ambitious climate goals, to keep the global warming under 2°C compared to the pre-industrial era, with an intent to limit it to 1.5°C. One of the main measures to be taken is the reduction of the use of fossil fuels.

For this reduction, car users should be stimulated to use public transport, but public transport itself also has to take drastic measures. The fleets of vehicles used have to be made more environmentally friendly. The way to achieve this is by replacing old vehicles with cleaner ones. The fastest way would be to replace all vehicles with electric vehicles at once, but this is not realistic because of financial and logistic reasons. A shift towards cleaner vehicles comes with a *transition phase* where we have to employ a heterogeneous fleet with a variable number of diesel vehicles with different exhaust characteristics and electric vehicles.

In this chapter we will present our research on vehicle scheduling during the transition phase towards cleaner vehicles. We will discuss two scenarios: one in which the fleet consists of a mixture of diesel vehicles with different exhaust characteristics, and one in which the fleet consists of two types of vehicles: diesel vehicles and electric vehicles. We show that by enhancing existing vehicle scheduling models in a relatively easy way, a significant contribution to a cleaner environment can be achieved.

After discussing the literature in Section 6.2, we present our model and results on scheduling a vehicle fleet with multiple exhaust characteristics in Section 6.3. Using a real-life case from Utrecht in the Netherlands, we show that taking these characteristics into account may save up to 20 percent of exhaust at almost no cost, while it is also possible to reduce the pollution by public transport even further in the city center. In Section 6.4 we discuss the intermediary steps when going from a 100 percent diesel fleet towards a 100 percent electric fleet. We show that by using a sophisticated algorithm to determine the best vehicle schedule for a mixed fleet, we obtain better solutions than starting with a full electric vehicle schedule

and straightforwardly replacing electric vehicles with diesel vehicles until the number of electric and diesel vehicles matches the number of available vehicles. We end this chapter with a conclusion in Section 6.5.

6.2 Literature overview

As we can read already in Gwilliam et al. [2004], air pollution is getting a big problem worldwide. Especially in cities, air pollution has serious consequences for public health. Important causes of air pollution are industry and transportation. For a long time, governments are taking measures to reduce pollution. This is done in two ways: by offering fiscal benefits in order to motivate improvement, and by making legislation in order to force improvement.

A lot of research has been done on the area of sustainable public transport. There are several studies on the properties of vehicles, which try to identify which kind of vehicles are the most sustainable and cost-effective. A comparison of the various alternative energy sources for public transport is described in Tzeng et al. [2005]. Lindgren [2015] compares the energy cost and total cost of hybrid and electric vehicles, while Nurhadi et al. [2014] and Lajunen [2014] have made a thorough analysis of the use of electric vehicles in medium-sized Swedish cities. However, they do not consider the phase during the transition where the fleet of vehicles consists of multiple types of vehicles. In Li et al. [2015], the transition towards a more environment-friendly vehicle fleet is discussed. Pollution is expressed in money, and the optimal replacement scheme is determined. However, in their article the distance driven by a vehicle is set to the average of all vehicles, while it would be more useful to make use of the cleaner vehicles as much as possible.

On vehicle scheduling during the transition phase, we found only one article, Reuer et al. [2015], in which the intermediate situation where there are both electric and diesel vehicles is studied. A vehicle schedule is created, without taking the type of vehicle into account. Given this vehicle schedule, the percentage of EVs that can be used is determined. This is done by testing for all vehicle tasks whether an EV is capable of driving it, combined with making small changes in order to increase the number of vehicle tasks suitable for an EV.

6.3 Environment-friendly vehicle scheduling

In this section, we discuss a model in which we take air pollution into account during vehicle scheduling, using a realistic fleet with multiple types of diesel vehicles. Recall that electric vehicles have not been taken into account yet; the transition towards electric vehicles will be discussed in Section 6.4. We also look at possibilities to reduce the pollution even more in 'black spots', for example city centers which are already heavily polluted. We also look for a balance between operational cost and pollution. We will illustrate our model with a real-life case.

In Section 6.3.1, we give some background information about European emission standards. Then we first give a discussion of methods to obtain schedules that limit the exhaust of greenhouse gases in Section 6.3.2, after which we define the ENVIRONMENT-FRIENDLY

VEHICLE SCHEDULING PROBLEM and propose a model to solve it, using the generic vehicle scheduling model that is described in Section 6.3.3. After that, we will give a description of the real-life case that we study in Section 6.3.4. The results of the environment-friendly vehicle scheduling are shown in Section 6.3.5. We finish with a conclusion in Section 6.3.6.

6.3.1 European Emission Standards

As mentioned earlier, governments may put legislation in place in order to reduce pollution. For the exhaust gases from combustion engines, the European Union imposes limits on exhaust for new vehicles sold. These limits are regularly tightened in order to move along with the technical possibilities and to challenge and motivate engine manufacturers to make engines more environment-friendly. Different limits are defined for cars, light commercial vehicles, trucks and buses. On website ECOpoint [2018], a lot of information about these European Emission Standards can be found. We will summarize the most important and relevant details here.

The limits are imposed on the measurements of a few harmful exhaust gases: carbon monoxide (CO), hydrocarbons (HC), nitrogen oxides (NO_x) and particulate matter (PM). In Table 6.1, the European Emission Standards since 1992 are listed, ordered by strictness of the requirements. Between Euro V and Euro VI we see an intermediate step called Euro EEV, where EEV is the abbreviation of Enhanced Environmentally friendly Vehicle. The differences in the limits make clear that there is a large difference in exhaust between vehicles that only differ a few years in age. For example, when we compare a vehicle from 2010 complying to emission norm Euro V with a 5 years younger vehicle from 2015 complying to emission norm Euro VI, the allowed HC exhaust is divided by 3, the NO_x exhaust is divided by 5 and the PM exhaust is halved. The observation that the exhaust limits decreased rapidly in this period will be used throughout this chapter.

Emission norm	Year of enforcement	CO (g/kWh)	HC (g/kWh)	NO_x (g/kWh)	PM (g/kWh)
Euro I	1992	4.5	1.10	8.0	0.36
Euro II	1996	4.0	1.10	7.0	0.15
Euro III	2000	2.1	0.66	5.0	0.10
Euro IV	2005	1.5	0.46	3.5	0.02
Euro V	2008	1.5	0.46	2.0	0.02
Euro EEV	1999	1.5	0.25	2.0	0.02
Euro VI	2014	1.5	0.13	0.4	0.01

Table 6.1: European Emission Standards for trucks and buses

6.3.2 Means to reduce air pollution by exhaust gases

There are many possible ways to achieve a reduction in air pollution by exhaust gases of vehicles with combustion engines. Some of them are:

- Reduce fuel consumption
- Reduce exhaust by changing driving behavior
- Use vehicles with less exhaust
- Use fuel that burns cleaner

The first two items are closely related, because the exhaust is dependent on the amount of fuel used. Fuel consumption can be reduced by reducing speed and by smoother acceleration and reducing braking. For reduction of exhaust, we observe that the exhaust of for example Particulate Matter is dependent on the speed that the engine is running. Keeping the engine running at optimal speed can be achieved by changing the transmission from motor to wheels or tuning the automatic transmission, if applicable.

Using cleaner vehicles is fairly straightforward: use the cleanest vehicles as much as possible, or replace vehicles with cleaner ones. Using cleaner fuel may be cheaper, when a new fuel is chosen that is compatible with the vehicles (or can be made compatible cost-efficiently).

In the next section we describe the way we model vehicle scheduling in general and we propose some modifications to this model in order to take air pollution into account.

6.3.3 Model used for environment-friendly vehicle scheduling

We model the ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM as an INTEGER LINEAR PROGRAMMING problem. The ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM is defined as an extension of the VEHICLE SCHEDULING PROBLEM, where minimizing air pollution is part of the objective of the problem.

We start with the underlying VEHICLE SCHEDULING PROBLEM, which we model as a multi-commodity network flow model. For every combination of depot and vehicle type, we create a *graph* in which we include all trips that are allowed for this combination of depot and vehicle type. By creating a graph for every combination of depot and vehicle type we model that every vehicle returns to the depot where it started its vehicle task. In Figure 6.1, we have drawn an example of such a graph. In this graph, representing one combination of depot and vehicle type, every trip is represented by a node (the rectangles in the figure) and every possible time that a bus may enter or leave the depot is also represented by a node (the circles at the bottom of the figure). Between these nodes, we create the following types of arcs:

- Between two trip nodes, we create an arc when these two trips are allowed to be consecutive. The variable assigned to this arc will get value 1 or 0, depending on the use of this arc.

- Between a trip node and a depot node, we create an arc when this is a valid pull-out or pull-in trip. The variable assigned to this arc will get value 1 or 0, depending on the use of this arc.
- Between two consecutive depot nodes, we create an arc to represent the number of vehicles that stay at the depot between the two times that are represented by the nodes. The variable assigned to this arc will get a non-negative, integral value.
- From the last depot node to the first depot node, we create an arc to assure that the number of vehicles of one type at one depot will be the same at the start and at the end. The variable belonging to this arc represents the number of vehicles that are parked at the depot overnight.

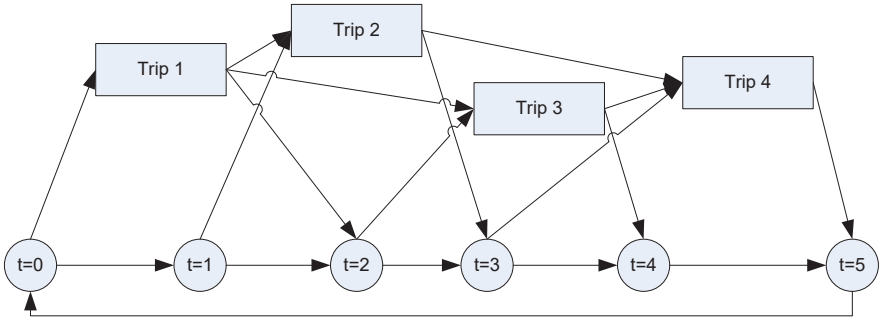


Figure 6.1: Graph for vehicle scheduling, representing one vehicle type for one depot. The round circles at the bottom represent depot nodes, the rectangles represent the trip nodes representing the trips allowed for this depot and vehicle type.

For every node T_{idv} representing trip i on depot d driven with vehicle type v, we represent using the combination of depot d and vehicle type v for trip i with variable n_{idv} where $n_{idv} \in \{0, 1\}$. For every node D_{dvt} on depot d for vehicle type v at time t with $t \in \mathbb{N}$, we represent the number of vehicles in the depot at time t with p_{dvt} where $p_{dvt} \in \mathbb{N}$.

For every arc $A_{T_1T_2}$ from a trip node T_1 to another trip node T_2, we represent the use with $a_{T_1T_2}$, where $a_{T_1T_2} \in \{0, 1\}$. We denote the cost of an arc $A_{T_1T_2}$ as $c_{T_1T_2}$. Similarly, the use of the arcs $A_{T_1D_2}$, $A_{D_1T_2}$ and $A_{D_1D_2}$ from trip node T_1 or depot node D_1 to trip node T_2 or depot node D_2 is represented with $a_{T_1D_2}$, $a_{D_1T_2}$ and $a_{D_1D_2}$, where $a_{T_1D_2} \in \{0, 1\}$, $a_{D_1T_2} \in \{0, 1\}$ and $a_{D_1D_2} \in \mathbb{N}$ and their costs by $c_{T_1D_2}$, $c_{D_1T_2}$ and $c_{D_1D_2}$. We assign the real variable cost per hour or per distance to all arcs that represent driving. For the arcs that represent parking at a depot, we do not assign cost, except for the arc that represents the number of vehicles at night. To this arc, we assign the fixed cost per vehicle. In Section 6.3.4, when we extend the model to solve the ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM and take air pollution into account, we will change these $c_{T_1T_2}$, $c_{D_1T_2}$ and $c_{T_1D_2}$ values by an appropriate amount.

The formulation as an ILP is as follows:

$$\text{Minimize} \quad \sum_{T_1,T_2} c_{T_1T_2}a_{T_1T_2} + \sum_{T_1,D_2} c_{T_1D_2}a_{T_1D_2} + \sum_{D_1,T_2} c_{D_1T_2}a_{D_1T_2} + \sum_{D_1,D_2} c_{D_1D_2}a_{D_1D_2} \tag{6.1}$$

$$\text{Subject to} \quad \sum_{d,v} n_{idv} = 1 \qquad \forall i \tag{6.2}$$

$$\sum_{T_1} a_{T_1T_{idv}} = n_{idv} \qquad \forall T_{idv} \tag{6.3}$$

$$\sum_{T_2} a_{T_{idv}T_2} = n_{idv} \qquad \forall T_{idv} \tag{6.4}$$

$$\sum_{T_1} a_{T_1D_{dvt}} + \sum_{t'} a_{D_{dvt'}D_{dvt}} = p_{dvt} \qquad \forall D_{dvt} \tag{6.5}$$

$$\sum_{T_2} a_{D_{dvt}T_2} + \sum_{t'} a_{D_{dvt}D_{dvt'}} = p_{dvt} \qquad \forall D_{dvt} \tag{6.6}$$

$$\text{Where} \quad n_{idv} \in \{0,1\} \tag{6.7}$$

$$a_{T_1T_2} \in \{0,1\} \tag{6.8}$$

$$a_{T_1D_2} \in \{0,1\} \tag{6.9}$$

$$a_{D_1T_2} \in \{0,1\} \tag{6.10}$$

$$a_{D_1D_2} \in \mathbb{N}, a_{D_1D_2} \geq 0 \tag{6.11}$$

$$p_{dvt} \in \mathbb{N}, p_{dvt} \geq 0 \tag{6.12}$$

In Equation (6.2), we make sure that for every trip, exactly one node is chosen, representing the choice that is made for vehicle and depot. In Equation (6.3), we take care that for every trip-node, the number of incoming arcs is equal to the value of the node, that is 1 when the node is chosen and 0 otherwise. Equation (6.4) does the same for outgoing arcs. For depot-nodes, we do the same in Equations (6.5) and (6.6).

In the special cases of one depot and one vehicle type or multiple depots and one vehicle type where the number of vehicles at the start and at the end of the day is equal for every depot, the problem can easily be modeled as a minimum weight bipartite matching problem, where the starts and ends of every trip should be matched. Trip ends at the end of the day can be matched to trip starts at the begin of the day, representing an overnight parking at a depot. In Section 12.4, Ahuja et al. [1993] show that this special case is solvable in $\mathcal{O}(V^2E)$ time using the Hungarian algorithm, where V is the number of nodes and E is the number of arcs.[1]

In all other cases the problem is in fact a minimum-cost multi-commodity flow problem for integral flow, which is known to be NP-hard in general.

For the ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM, we modify the real cost for every trip or arc $c_{T_1T_2}$ in Equation (6.1) by a fictional cost that represents the

[1]The upper bound for the Hungarian algorithm can be improved to $\mathcal{O}(VE \log V)$ by using advanced data-structures. Kao et al. [2001] gave an algorithm for the problem that achieves an $\mathcal{O}(\sqrt{V}EW)$ bound, when all weights are non-negative, where W is equal to the largest weight.

exhaust. In our experiments we vary the extent of the cost modification. We solve the ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM by using a generic ILP-solver.

6.3.4 Case description: Utrecht

For the experiments we use the planning of the public bus transport in the region around Utrecht[2]. An important issue in this area was that the reduction in pollution of the environment had to be taken into account, especially from the exhaust gases from the buses. The total exhaust should be reduced, and the exhaust in a list of specified 'black spots' in the center of Utrecht should be reduced even more. Furthermore, the public transport company has to use 142 older buses that are more polluting than the newer buses.

The case study consists of public transport by tram and bus. In this section, we only consider the buses. The 3 electric buses that are available in this case are planned in the same way as diesel buses, charging is not taken into account. The main characteristics of the case are as follows:

- 5500 trips per workday
- 5 different vehicle types in use:
  - 10 Minibuses: small bus for up to 8 passengers
  - 3 Electric buses: bus with electric traction, for use on route 2
  - 197 Standard buses: 12-meter long bus. 131 of them are old buses, compliant with Euro EEV standard, 66 of them are new buses, compliant with Euro VI standard
  - 87 Articulated buses: 18-meter long bus, for the busier trips. 11 of them are old buses, compliant with Euro EEV standard, 76 of them are new buses, compliant with Euro VI standard
  - 17 Double-articulated buses: 25-meter long bus, for use on the busiest routes, compliant with Euro VI standard.
- In total 314 vehicles are available
- 1 depot is used for this case study

The properties of all vehicle types used can be found in Table 6.2. On top of this, we use a cost of 30 per hour of use of a bus to represent the cost for a driver. Without these driver costs, buses that are only used in the peak hours will be left at a stop between the peak hours

[2]This case study is based on a tender issued by the 'Bestuur Regio Utrecht'(BRU) in 2012 for the public transport in the municipality of Utrecht and eight municipalities around Utrecht. An important issue in this tender was that BRU required that the environment was taken into account, especially the exhaust gases from the buses. The total exhaust should be reduced, and the exhaust in a list of specified 'black spots' in the center of Utrecht should be reduced even more. Furthermore, BRU expected from the winning company to use the 142 old buses built in 2008 that were in service in the area. These buses complied with the European Euro EEV standard for exhaust.

because driving to and from the depot is more expensive than parking at a stop for a few hours.

In Table 6.2, there are no values about exhaust for minibuses and the cost values for the electric vehicles are set to be equal to the cost values for the standard vehicles. These assumptions will not influence the outcome, because these two vehicle types have their own set of trips which have only one allowed vehicle type and are not allowed on other trips than their own.

	Fixed cost per veh	Var cost per km	CO_2 g/km	PM g/km	NO_x g/km
Dbl Articulated	315.13	1.15	1733	3.6	576
Articulated Euro VI	244.13	0.91	1179	9.2	571
Articulated EEV	244.13	0.91	1271	33.6	4082
Standard Euro VI	133.00	0.66	892	1.9	296
Standard EEV	133.00	0.66	909	24.5	2889
Electric Vehicle	133.00	0.66	0	0.0	0
Minibus	59.26	0.31			

Table 6.2: Parameters used in vehicle scheduling

Our objective is to minimize exhaust in a cost-efficient way. We will do this by solving the ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM, where we set the fictional cost to a value that is proportional to the amount of PM that is exhausted. We use PM because this figure is important for Utrecht, because of European guidelines of PM concentration in urban areas. Another objective mentioned before is reducing the exhaust in the 'black spots' in the city center. For this, we apply an additional fictional cost when an older vehicle passes the bus lane through the center of Utrecht. The exact cost used for these fictional costs is varied among the experiments.

6.3.5 Computational results

We implement the model from Section 6.3.3 and start with an optimization of our case study of Utrecht where we do not take air pollution into account. We use the same cost for the Euro VI and the Euro EEV vehicles. So in this first optimization we can combine all articulated buses to one type and all standard buses to one type. After this, we run two series of optimizations where in both we added extra cost equal to the exhaust of PM multiplied by a factor increasing at every step. In the second series of optimization we also applied a cost of 1 for every Euro EEV bus that passes the bus lane through the city center. The first line in both series (with factor 0) is the situation where we do not take exhaust figures into account. The results can be found in Tables 6.3 and 6.4.

In Table 6.3, we see that increasing the PM factor from 0 to 0.005 has a large impact on PM and NO_x exhaust. So by just taking exhaust into account, we can reduce the exhaust by 10 percent at no cost. The CO_2 exhaust stays the same, because CO_2 exhaust is mainly dependent on fuel consumption, which is about the same for Euro EEV and Euro VI buses.

With a further increase of the factor, we see that the exhaust is further diminished, but the cost without the fictional cost for exhaust increases. We also see that the number of buses is increasing. This turned out to be caused by the addition of extra Euro VI buses, while some Euro EEV buses are still part of the fleet, have their fixed cost contributing to the total cost, but are planned to stand still.

When we compare Table 6.3 to Table 6.4, we see that taking passage of the center into account does not change the solution very much. Only the solution with factor 0, where we do not take PM exhaust into account, shows a large difference in exhaust. This is because using only the passage of the center is also a measure of pollution in the whole area, as most trips pass this center.

In Figure 6.2, we have mapped the results from the run with an additional cost of 0.005 for every kg PM exhaust and an additional cost of 1 for every trip that passes the city center. The width of a line indicates the number of buses that drive through that road per day, the color shows the reduction of PM exhaust per day compared to the situation where we only use Euro EEV buses.

PM factor	Total cost	Vehicles	CO_2 kg	PM kg	NO_x kg
0.000	205144.2	279	69.3	1150.8	131.2
0.005	205248.7	279	69.1	969.4	109.5
0.010	205518.5	279	69.3	931.8	104.7
0.015	206736.2	287	69.3	843.3	94.4
0.020	211593.5	324	69.1	569.6	63.1
0.050	219665.6	386	68.6	292.0	31.9
0.100	222517.8	405	68.4	251.1	26.9

Table 6.3: Results from optimization with additional cost on PM exhaust

PM factor	Total cost	Vehicles	CO_2 kg	PM kg	NO_x kg
0.000	205220.8	279	69.2	1048.0	118.7
0.005	205364.6	279	69.3	965.7	108.9
0.010	205792.5	280	69.4	916.8	102.8
0.015	207738.7	294	69.4	786.7	87.8
0.020	213049.3	335	69.0	502.2	55.5
0.050	219665.6	386	68.6	292.0	31.9
0.100	222517.8	405	68.4	251.1	26.9

Table 6.4: Results from optimization with additional cost on PM exhaust and driving through the center

We solved all vehicle scheduling problems using CPLEX 12.2 on a desktop computer

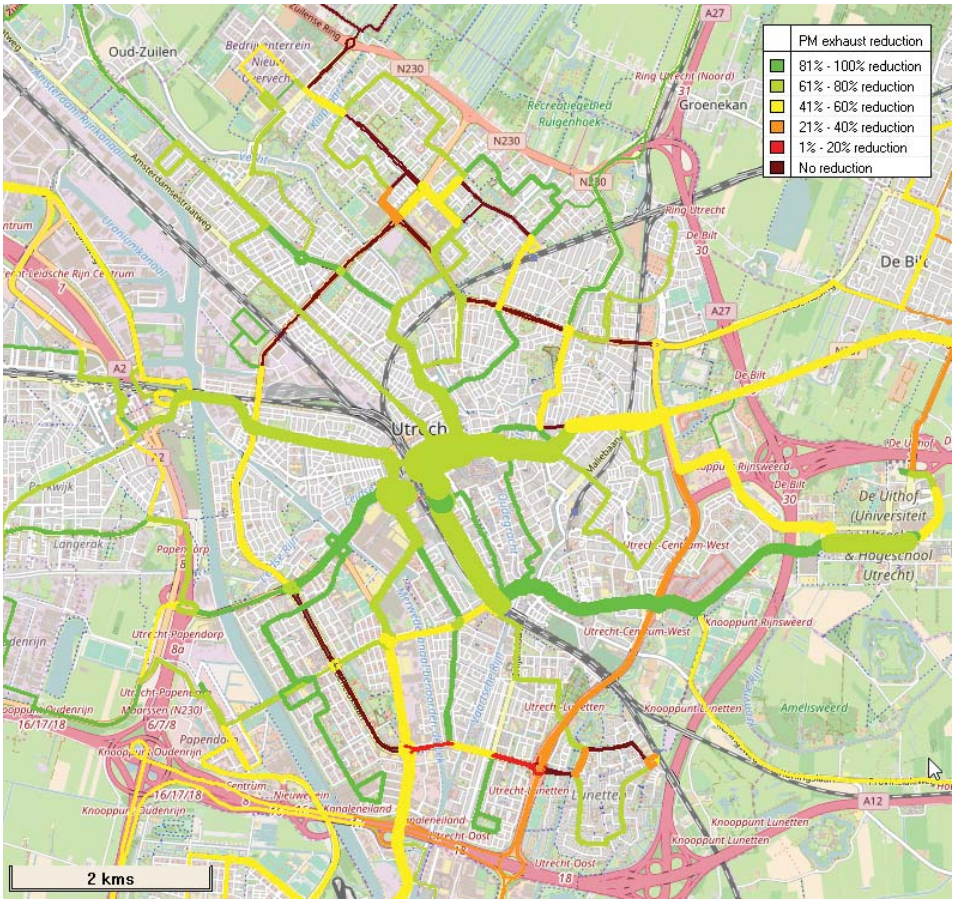


Figure 6.2: Map indicating the reduction of PM exhaust related to the situation with only Euro-EEV-buses. The width of a line corresponds with the number of buses per day.

with an Intel®Core™i7-3770 CPU running on 3.4 GHz and equipped with 16 GB of RAM memory. The problems are all solved to optimality in about 12 minutes.

6.3.6 Conclusion

We looked at scenarios where we have a mixed fleet of old and new vehicles and we aim at minimizing the exhaust. For this problem, we defined the ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM, which we model as a multi-commodity network flow model. We assign additional virtual costs for exhausts to the trips and solve the model using an INTEGER LINEAR PROGRAMMING formulation, which was solvable to optimality in 12 minutes. We have shown in a case study that by applying additional cost for exhaust, we can reduce the exhaust of PM and NO_x typically by 20 percent at almost no cost. Further reduction of the exhaust is possible by replacing the older Euro EEV buses with newer Euro VI ones, but this is an expensive solution.

Reduction of PM exhaust in black spots also gives good results at almost no cost. We have plotted the results of our optimization on a map in Figure 6.2, where we clearly see that for our case, the exhaust of PM is reduced in the whole city of Utrecht, but even more in the city center, where the black spots are.

6.4 Transition towards electric vehicles

6.4.1 Introduction

When the vehicles of a public transport system are to be replaced by EVs, this is usually not done in a single step for several reasons. First of all, the cost of a sudden change to many EVs can be prohibitive. Another reason can be that switching to new technology may be considered too risky. It means that, in general, a gradual transition to EVs is preferred.

In this section, we will discuss the scheduling problems that are encountered when we schedule vehicles for public transport, given that we have two types of vehicles available: electric vehicles and diesel vehicles, and that a mixture of both should be used where the number of available diesel vehicles is fixed and where EVs are usually preferred because they produce no exhaust gases. For solving this scheduling problem, we use two methods. In Section 6.4.2.1 we introduce Method I, where we first schedule everything for EVs and then replace EVs in the schedule with diesel vehicles in order to match the number of each kind of vehicle in the fleet. In Method II we take the properties for EVs and diesel vehicles into account while scheduling.

We will describe the model for Method II as an INTEGER LINEAR PROGRAMMING problem in Section 6.4.2.2. This model will be tested on two different datasets with different characteristics in Section 6.4.3, after which we will show in Section 6.4.4 that our algorithm gives better results than Method I, the stepwise approach in which we first schedule everything for EVs and then replace the vehicles on the shorter blocks with diesel vehicles in order to match the available number of EVs.

Electric vehicle:	133.00	each
Diesel vehicle:	133.00	each
Time cost:	30.00	per hour
Average speed:	20.00	km per hour
Driving cost electric vehicle:	0.13	per km
Driving cost diesel vehicle:	0.66	per km

Table 6.5: Costs used to determining vehicle schedule cost.

6.4.2 Models for transition towards EVs

6.4.2.1 Method I

In Method I, we start with the solution for an all electric fleet. Because every vehicle task for an electric vehicle can also be driven by a diesel vehicle, we can gradually replace the electric vehicles by diesel vehicles. We use a greedy algorithm to determine in which order the electric vehicles are replaced. We start with replacing the electric vehicle task with the shortest driving time with a diesel vehicle, in order to get the cheapest solution with one diesel vehicle. We continue replacing the electric vehicles until all vehicle tasks are driven by a diesel vehicle.

6.4.2.2 Method II

For Method II, we will use an adaptation of Model 1c which we described in Section 5.5.4. The earlier described Model 1c models the scheduling of electric vehicles for one vehicle type and one depot. Here, we want to schedule electric vehicles as well as diesel vehicles, so we have to extend the model to handle this. We create an *extra graph* for diesel vehicles in the same way as described in Section 5.5.1. This diesel graph is linked to the electric graph by requiring that for every trip, only one node is chosen, either in the diesel graph or in the electric graph.

We solve the model using COLUMN GENERATION with LAGRANGIAN RELAXATION as described in Section 5.5.4. We add the shadow prices of all constraints to the corresponding trip. In this way every path in the graph represents a valid vehicle task for a vehicle and the total cost of that path is the reduced cost of that column. For the electric graph we determine the shortest path using the label-correcting algorithm as described in Section 5.5.4 and add this shortest path to the RMP. For the diesel graph we use a standard shortest-path algorithm to determine the column to be added to the RMP. For the cost we use the figures from Table 6.5.

6.4.3 Computational results

For the evaluation of our model, we apply Methods I and II and compare the results from both methods. We use the full dataset that is also used in Section 4.8 in combination with electric vehicles with a battery capacity of 122 kWh and 244 kWh. When we solved this model, we

		Method I					Method II							Diesel	
# Vehicles Diesel	# Vehicles Electric	Drv time Diesel	Total time Diesel	Drv time Electric	Total time Electric	Total Cost	Drv time Diesel	Total time Diesel	Drv time Electric	Total time Electric	Total Cost	Cost improvement		Distance Reduct.	Exhaust Improv.
0	29	0:00	0:00	354:16	422:07	17441.59	0:00	0:00	354:16	422:07	17441.59	0.00	0.0%	0	
1	27	9:38	10:50	342:33	399:11	17543.71	7:33	8:27	344:38	401:34	17020.21	523.50	3.0%	42	21.63%
2	26	19:18	22:20	335:11	387:41	17646.17	33:55	37:08	320:34	372:53	17172.67	473.50	2.7%	-292	-75.73%
3	25	29:28	34:55	321:15	371:13	17753.94	52:47	59:06	297:56	347:02	17246.37	507.57	2.9%	-466	-79.13%
4	24	40:13	48:04	312:55	358:06	17867.89	39:01	44:12	314:07	361:58	17240.72	627.17	3.5%	24	2.98%
5	23	51:02	60:22	299:31	344:12	17982.55	63:26	70:49	287:07	333:45	17311.82	670.72	3.7%	-248	-24.30%
6	22	61:56	72:30	288:28	333:15	18098.09	72:55	82:42	277:29	323:03	17447.46	650.63	3.6%	-220	-17.73%
7	21	72:56	85:07	279:29	323:49	18214.69	71:59	81:02	280:26	327:54	17671.31	543.38	3.0%	19	1.30%
8	19	84:13	98:23	267:22	305:24	18334.29	97:09	108:09	254:26	295:38	17648.41	685.88	3.7%	-259	-15.36%
9	18	95:30	110:34	256:28	291:37	18453.89	107:59	119:48	243:59	282:23	17716.24	737.66	4.0%	-250	-13.07%
10	17	106:50	123:19	244:41	280:16	18574.03	120:27	135:09	231:04	268:26	17889.21	684.81	3.7%	-272	-12.75%
11	16	118:12	136:56	234:30	264:06	18694.51	139:57	156:10	212:45	244:52	18022.49	672.02	3.6%	-435	-18.40%
12	16	129:51	150:15	220:40	249:19	18818.00	147:21	165:40	203:10	233:54	18051.25	766.75	4.1%	-350	-13.48%
13	14	141:47	163:35	210:27	235:59	18944.50	161:06	179:13	191:08	220:21	18201.47	743.03	3.9%	-386	-13.62%
14	12	153:47	178:16	199:23	221:59	19071.70	173:02	195:14	180:08	205:01	18350.89	720.81	3.8%	-385	-12.52%
15	12	165:52	193:03	186:40	205:35	19199.78	179:37	200:33	172:55	198:05	18370.52	829.26	4.3%	-275	-8.29%
16	11	178:19	207:45	172:40	189:21	19331.75	195:24	219:41	155:35	177:25	18487.80	843.95	4.4%	-342	-9.58%
17	10	191:01	223:13	164:01	181:32	19466.37	213:16	244:43	141:46	160:02	18917.21	549.16	2.8%	-445	-11.65%
18	9	203:51	237:36	148:42	159:14	19602.40	226:33	254:50	126:00	142:00	18814.06	788.34	4.0%	-454	-11.14%
19	8	216:58	253:11	136:08	143:00	19741.44	244:54	274:17	108:12	121:54	18990.50	750.94	3.8%	-559	-12.87%
20	7	230:13	268:38	124:21	130:10	19881.89	255:13	286:23	99:21	112:25	19182.17	699.72	3.5%	-500	-10.86%
21	6	244:06	284:00	108:01	109:01	20029.05	272:58	303:05	79:09	89:56	19190.45	838.60	4.2%	-577	-11.83%
22	5	258:16	300:26	98:06	94:25	20179.22	284:20	315:38	72:02	79:13	19509.99	669.23	3.3%	-521	-10.09%
23	4	272:47	317:31	83:38	79:25	20333.10	304:38	338:23	51:47	58:33	19654.80	678.30	3.3%	-637	-11.68%
24	4	287:24	333:29	67:04	61:16	20488.03	313:26	349:16	41:02	45:29	19677.51	810.53	4.0%	-521	-9.06%
25	2	302:06	350:34	53:00	42:30	20643.85	330:28	365:52	24:38	27:12	19809.21	834.65	4.0%	-567	-9.39%
26	1	317:03	367:57	40:07	30:28	20802.32	329:20	366:37	27:50	31:48	19963.07	839.26	4.0%	-246	-3.87%
27	0	348:13	386:48	0:00	0:00	19791.46	348:13	386:48	0:00	0:00	19791.46	0.00	0.0%	0	0.00%

Table 6.6: Results for transition path optimization from diesel vehicles to electric vehicles, using EVs with 122 kWh battery capacity. For every number of diesel vehicles, the results of Method I and II can be found the table, followed by the difference in cost and the difference in distance driven by diesel vehicles.

got a solution with only electric vehicles. When we only allow diesel vehicles by removing the electric part of the graph, we get a slightly more expensive solution. In order to get the intermediate scenarios we have to force the use of diesel vehicles. We do this by adding a 0-minute dummy trip only to the diesel-graph at 2:00 in the morning, one for every required diesel vehicle. We start with 1 diesel vehicle and continue until all trips are driven by diesel vehicles.

In order to evaluate the results of Method II described in Section 6.4.2.2, we have also tested Method I. The results are compared with the results from Method II described in Section 6.4.2.2. The results for the EV with a battery capacity of 122 kWh can be found in Table 6.6, the results for 244 kWh battery capacity can be found in Table 6.7. In these tables, the first column contains the number of diesel vehicles that have to be used, the number of EVs results from the optimization. The column with Cost improvement contains the difference between the total cost of Method I and Method II; it shows by what extent Method II is cheaper than Method I. In the last two columns, the difference in distance driven by diesel vehicles is calculated, which is proportional to the exhaust.

The total cost for Method I and Method II is compared in Figure 6.3 for the EV with 122 kWh battery capacity and in Figure 6.4 for the EV with 244 kWh battery capacity.

6.4.4 Conclusion

When we compare Method II with Method I to determine vehicle schedules for a fleet of electric and diesel vehicles, we see that for EVs with a battery capacity of 122 kWh, Method

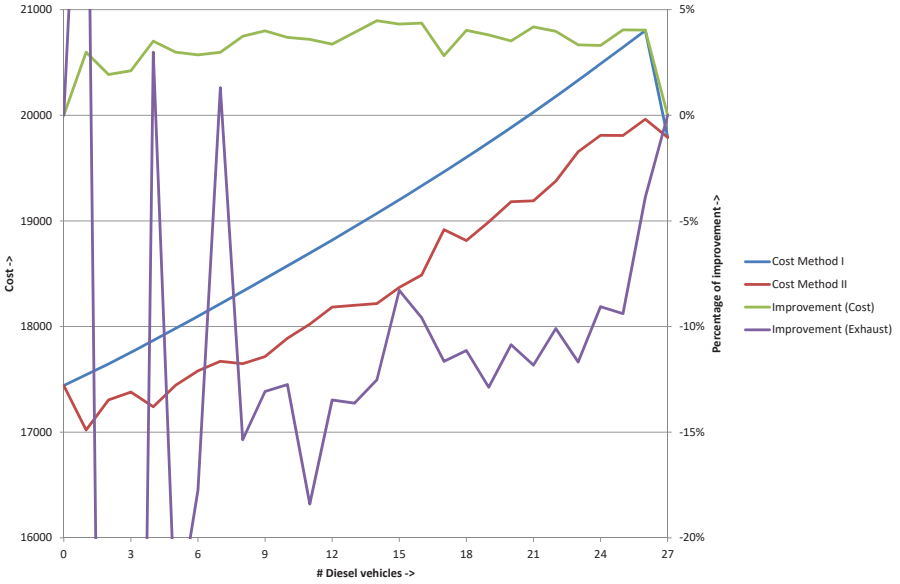


Figure 6.3: Cost comparison for transition from electric to diesel, using EVs with 122 kWh battery capacity

		Method I					Method II							Diesel	
# Vehicles Diesel	# Vehicles Electric	Drv time Diesel	Total time Diesel	Drv time Electric	Total time Electric	Total Cost	Drv time Diesel	Total time Diesel	Drv time Electric	Total time Electric	Total Cost	Cost improvement		Distance Reduct.	Exhaust Improv.
0	28	0:00	0:00	356:41	400:00	16651.38	0:00	0:00	356:41	400:00	16651.38	0.00	0.0%	0	
1	27	8:37	9:54	346:34	388:46	16742.71	8:52	10:04	346:19	388:36	16834.46	-91.75	-0.5%	-5	-2.90%
2	26	18:01	20:11	337:30	374:58	16842.35	15:27	16:40	340:04	378:29	16666.61	175.74	1.0%	51	14.25%
3	25	28:04	31:49	329:58	353:14	16948.88	29:48	32:04	328:14	352:59	16522.27	426.62	2.5%	-35	-6.18%
4	24	39:04	43:56	314:51	350:01	17065.48	31:21	34:49	322:34	359:08	16794.99	270.49	1.6%	154	19.75%
5	23	50:22	56:16	303:09	337:33	17185.26	45:59	51:46	307:32	342:03	16945.07	240.20	1.4%	88	8.70%
6	22	61:55	69:41	293:20	327:36	17307.69	58:24	65:36	296:51	331:41	17185.19	122.50	0.7%	70	5.68%
7	21	73:29	82:22	278:30	309:23	17430.30	64:07	72:30	287:52	319:15	17071.29	359.01	2.1%	187	12.75%
8	19	85:44	95:48	266:12	296:13	17560.15	66:54	74:50	285:02	317:11	16975.67	584.48	3.3%	377	21.97%
9	18	98:00	108:58	257:33	286:03	17690.18	93:35	104:03	261:58	290:58	17357.91	332.26	1.9%	88	4.51%
10	17	110:17	122:14	244:38	274:05	17820.38	109:43	124:32	245:12	271:47	17566.28	254.10	1.4%	11	0.51%
11	16	122:40	136:16	230:55	257:02	17951.64	117:02	131:13	236:33	262:05	17549.87	401.77	2.2%	113	4.59%
12	16	135:20	149:59	217:42	243:12	18085.91	141:52	156:46	211:10	236:25	17941.17	144.74	0.8%	-131	-4.83%
13	14	148:03	164:05	207:40	233:48	18220.71	151:16	170:30	204:27	227:23	18055.79	164.92	0.9%	-64	-2.17%
14	12	160:50	179:02	194:48	219:51	18356.21	159:33	181:32	196:05	217:21	18173.38	182.83	1.0%	26	0.80%
15	12	174:20	194:08	183:12	203:57	18499.31	180:18	200:32	177:14	197:33	18374.27	125.04	0.7%	-119	-3.42%
16	11	187:53	209:22	168:44	190:42	18642.94	195:21	219:31	161:16	180:33	18590.91	52.03	0.3%	-149	-3.97%
17	10	201:26	226:40	157:51	175:02	18786.57	201:25	225:17	157:52	176:25	18711.15	75.42	0.4%	0	0.01%
18	9	215:21	242:07	141:11	157:02	18934.09	225:15	254:00	131:17	145:09	18880.14	53.95	0.3%	-198	-4.60%
19	8	229:55	259:16	133:55	149:16	19088.49	223:57	252:23	139:53	156:09	19166.84	-78.34	-0.4%	119	2.60%
20	7	245:06	276:10	115:27	127:04	19249.44	235:59	266:16	124:34	136:58	19126.85	122.58	0.6%	182	3.72%
21	6	260:28	293:04	97:00	105:50	19412.32	264:28	296:46	93:00	102:08	19290.76	121.56	0.6%	-80	-1.54%
22	5	275:58	310:12	81:02	87:39	19576.62	280:19	312:19	76:41	85:32	19426.06	150.57	0.8%	-87	-1.58%
23	4	291:33	327:37	67:48	73:24	19741.81	304:19	341:15	55:02	59:46	19781.57	-39.76	-0.2%	-255	-4.38%
24	4	307:24	345:50	52:27	56:27	19909.82	315:14	351:32	44:37	50:45	20069.58	-159.77	-0.8%	-157	-2.55%
25	2	323:31	363:49	37:58	38:58	20080.65	316:29	353:06	45:00	49:41	19969.08	111.57	0.6%	141	2.17%
26	1	339:38	381:34	28:33	35:44	20251.49	345:50	391:51	22:21	25:27	20733.11	-481.62	-2.4%	-124	-1.83%
27	0	348:13	386:48	0:00	0:00	19791.46	348:13	386:48	0:00	0:00	19791.46	0.00	0.0%	0	0.00%

Table 6.7: Results for transition path optimization from diesel vehicles to electric vehicles , using EVs with 244 kWh battery capacity. For every number of diesel vehicles, the results of Method I and II can be found the table, followed by the difference in cost and the difference in distance driven by diesel vehicles.

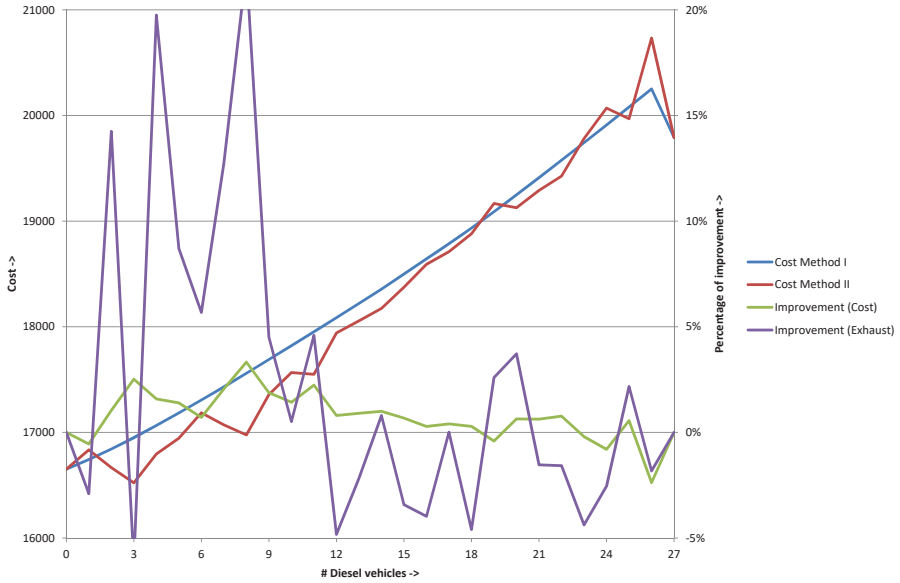


Figure 6.4: Cost comparison for transition from electric to diesel, using EVs with 244 kWh battery capacity

II gives solutions that are 3 to 4 percent cheaper than those obtained by Method I. However, we see that with respect to the total driving time for diesel vehicles, which gives a good representation of the level of exhaust, Method I gives better results than Method II. This could be caused by the fact that vehicle tasks for diesel vehicles have fewer constraints to comply to, so driving more trips with diesel can be cheaper because in this way some constraints can be avoided. As the cost reduction of 3 to 4 percent of Method II in comparison to Method I is quite small, the public transport company may choose to use the results from Method I because of the better exhaust figures and the flexibility of the result: extra EVs can be added to the fleet without the need of rescheduling.

When we compare the results for EVs with a battery capacity of 244 kWh, we see that Method II does not really reduce the cost compared to Method I, but the total exhaust is reduced drastically when using between 1 and 10 diesel vehicles. The constraints that lead to more rechargings during the day and thus more costs for EVs with a battery capacity of 122 kWh are here less restrictive because the EVs with a battery capacity of 244 kWh need fewer rechargings during a day.

6.5 General conclusion

In this chapter we have investigated some optimization problems that occur during the transition towards newer and cleaner vehicles in public transport.

We started with the transition from older to newer, cleaner diesel buses. We looked at scenarios where we have a mixed fleet of old and new vehicles and we aim at minimizing the exhaust. For this problem, we defined the ENVIRONMENT-FRIENDLY VEHICLE SCHEDULING PROBLEM, which we model as a multi-commodity network flow model. We assign additional virtual costs for exhausts to the trips and solve the model using an INTEGER LINEAR PROGRAMMING formulation. We have shown in a case study that by applying additional virtual cost for exhaust, we can reduce the exhaust of PM and NO_x typically by 20 percent at almost no real cost.

The second part of this chapter deals with the transition from diesel to electric vehicles. We look at vehicle scheduling using a mixed fleet of electric and diesel vehicles, which was modeled using two methods. Method I starts with a vehicle schedule where we first schedule everything for EVs and then replace EVs in the schedule with diesel vehicles in order to match the number of each kind of vehicle in the fleet. In Method II we use a multi-commodity network flow model, where the layers representing the EVs are modeled as in Chapter 5.

For our comparison of Method I with Method II, we have run Method II for EVs with a battery capacity of 122 kWh and 244 kWh and for all possible mixtures of EVs and diesel vehicles. We have also run Method I for comparison, where we take the full electric vehicle schedule and replace the EVs one by one with diesel vehicles without changing the vehicle schedule. We optimized these mixed scenarios for cost. We see that in the scenario with 122 kWh the solutions for mixed fleets from Method II are 3 to 4 percent cheaper than Method I, but we also see that the total exhaust is larger than when using Method I, despite the fact that the cost per kilometer for an EV is smaller than for a diesel vehicle. This could be caused by the fact that vehicle tasks for diesel vehicles have fewer constraints to comply to, so driving more trips with diesel can be cheaper because in this way some constraints can be avoided.

When we compare the results for the 244 kWh EVs, we see that there is no significant cost saving, but Method II gives results where the total exhaust is reduced drastically when the minority of the vehicles are diesel vehicles. This is caused by the fact that the larger battery leads to less restrictive constraints on the EVs with respect to recharging.

Chapter 7

Conclusions

We started this thesis with the observation that the goal of optimization of public transport is to make it fast, easy, reliable, efficient and sustainable. A lot of research has been done on public transport optimization, but since the availability of large amounts of measurements of runtimes and the increasing concerns of the environment, new questions arise.

In Chapter 3 we use the large amount of measurements of runtimes to improve the reliability of trip run times, in order to make the passing time of buses at the stops along a route more reliable. The reliability of vehicle tasks is improved further in Chapter 4 by using these measurements and by using the realized trip runtimes instead of the planned trip runtimes.

The increasing environmental concerns led to the introduction of electric vehicles, which come with additional constraints in the scheduling, because of the limited battery capacity. In Chapter 5 we introduce an efficient way to schedule electric vehicles. Because it is practically and financially infeasible to switch all vehicles in public transport to a more environment-friendly type at once, there is a transition phase in between, in which all vehicles are gradually replaced by cleaner ones. In Chapter 6 we show that for an optimal environmental effect, we have to take the properties of all vehicle types in a mixed fleet into account.

7.1 Determination of planned trip runtimes

We started in Chapter 3 with the definition of the TRIP RUNTIME DETERMINATION PROBLEM, which is the determination of planned trip runtimes on the basis of measured trip runtimes. We proposed several new methods and compared these to each other and to the well-known percentile-method, which is nowadays widely used. We distinguish methods that require runtimes to be integral and methods providing continuous runtimes. In the latter, runtimes can be rounded.

Our conclusion was that our method based on continuous optimization significantly outperforms the percentile-method: the average delay in the computed schedules allowing continuous runtimes can be reduced with about 60%. We also concluded that in almost all cases this method gives the best results in reasonable time. When we want integral departure times, then it is best to round down the fractional departure times found using the method of

continuous optimization.

The determination of planned trip runtimes is usually done in order to maximize punctuality or to minimize average delays on stops. Combining the trip runtime measurements with passenger counts makes it possible to optimize on average delay over all passengers and on average travel duration. Using the average delay over all passengers instead of over all stops makes busy stops more important. What we saw, was that optimization on passenger travel time instead of on average delay leads to a reduction in travel time of on average 4%, at the cost of punctuality and average delay on stops. When we use a combination of average travel time and average delay over all passengers as an objective, where both parts have equal weight, we get a reduction in travel time of about 3.5% at the cost of punctuality, but with minor impact on average delay on stops.

In our research we have restricted ourselves to one route, in order to be able to compare our methods with known methods from literature. Determining planned trip runtimes for a whole network may be useful, so we formulate:

Open Problem 1. *Solve the* TRIP RUNTIME DETERMINATION PROBLEM *for multiple routes simultaneously, especially when passengers change between these routes.*

7.2 Robust vehicle scheduling

After having investigated the stochastic nature of trip runtimes in Chapter 3, we continued with research on the stochastic nature of trip durations in vehicle scheduling. Instead of using a fixed trip duration in vehicle scheduling we used a trip duration distribution for every trip, which turned out to be dependent on the actual delay at the start of a trip. Using this information, we first developed a method to evaluate vehicle schedules for punctuality by representing the stochastic nature of the trip duration with a transition matrix. After this, we proposed and evaluated several methods to solve the STOCHASTIC DEPARTURE TIME DEPENDENT VEHICLE SCHEDULING PROBLEM.

Our research leads to the following conclusions:

- All methods with a predetermined minimum layover perform worse than the methods that add a fictional cost to the layover depending on the punctuality. We see that for the same exploitation cost, the punctuality of the former is about two to three percent better than the latter and the average delay is reduced with 30 to 40 percent.
- Allowing negative layovers between two trips increases the punctuality by one percent in the cheaper and least robust scenarios. Because these scenarios are generally not desirable, it is not useful to relax the constraint that layovers should be non-negative.
- It is not useful to take delay propagation into account over a sequence of more than two trips.

In Chapter 4, we looked at building vehicle schedules from a given timetable, using the measured trip runtime distributions. This subject may be combined with the open problem from Section 7.1:

Open Problem 2. *Include the robustness of the vehicle schedule in the objective of solving the* TRIP RUNTIME DETERMINATION PROBLEM *for multiple routes simultaneously.*

7.3 Scheduling electric vehicles in public transport

In Chapter 5 we have investigated the scheduling of electric vehicles (EVs) in public transport. We have shown that the properties of EVs, especially the limited range due to limited battery capacity, have to be taken into account, because otherwise we will end up with an infeasible vehicle schedule. For scheduling EVs, we developed methods to solve the E-VSP, and we saw that in most cases, extra vehicles are needed for a feasible vehicle schedule.

We started with a model based on INTEGER LINEAR PROGRAMMING, which handles all electricity usage and battery charging for the EVs during a day. This model gives the optimal solution for the E-VSP. The downside of this model is the long running time on medium and large datasets. In order to solve the E-VSP for larger datasets, we augmented the underlying graph with information about the State of Charge of the battery during the day. We also developed a solution method for both models based on COLUMN GENERATION and LAGRANGIAN RELAXATION. When we combined the augmented graph with solution methods based on COLUMN GENERATION and LAGRANGIAN RELAXATION, we were able to solve the E-VSP for real-life sized instances in reasonable time.

Our models are able to handle more properties of EVs and charging than we have investigated. This leads to:

Open Problem 3. *Our models offer the possibility to take time-of-day pricing and battery deprecation into account. Run experiments and evaluate the usefulness.*

Open Problem 4. *Consider more possibilities for charging place and time between two trips; now it is assumed that charging always takes place after a deadhead, but it may be beneficial to charge before the deadhead or even at an intermediary place.*

Furthermore, our models lack one major feature:

Open Problem 5. *Enhance our models with the possibility to set constraints on the maximum number of simultaneous chargings at a charging station.*

7.4 Modeling the transition towards sustainable public transport

In Chapter 6 we describe techniques that can be used when scheduling public transport during the transition phase towards a cleaner vehicle fleet.

In Section 6.3 we described a case study where we have a fleet with vehicles of different ages and different exhaust characteristics. We included the exhaust characteristics in our vehicle scheduling by assigning extra cost for exhaust. We showed in a case study that we could theoretically save 20 percent of the exhaust at almost no cost compared with vehicle scheduling without taking exhaust into account.

The second transition scenario is the transition from fossil fuel to electricity as energy provider, which is described in Section 6.4. We extended our model for solving the E-VSP, which is described in Chapter 5, with the possibility to schedule a mixture of electric vehicles and vehicles powered by fossil fuels. We saw that our Method II leads to solutions that are 3 to 4 percent cheaper than the simple Method I, which starts with a full electric vehicle schedule and replaces EVs with diesel buses as needed. However, we see that with respect to the total driving time for diesel vehicles, which gives a good representation of the exhaust, the simple method gives much better results than our Method II. This could be caused by the fact that vehicle tasks for diesel vehicles have fewer constraints to comply to, so driving more trips with diesel can be cheaper because in this way some constraints can be avoided.

We have investigated a mixed fleet with diesel vehicles with multiple exhaust characteristics, and a mixed fleet with diesel and electric vehicles. We did not look at mixed fleets with multiple types of EV, for example with different battery sizes or different charging characteristics. Even though this may not fit well in this chapter, the following may be worthwile to investigate:

Open Problem 6. *When there are multiple types of EVs with different characteristics available, what is the effect of using such a mixed fleet of EVs?*

Bibliography

J.D. Adler. *Routing and Scheduling of Electric and Alternative-Fuel Vehicles*. PhD thesis, Arizona State University, 2014.

R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-617549-X.

B. Amberg, B. Amberg, and N. Kliewer. Increasing delay-tolerance of vehicle and crew schedules in public transport by sequential, partial-integrated and integrated approaches. *Procedia Social and Behavioral Sciences*, 20:292–301, 2011.

C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46: 316–329, 1998.

R. Bellman. The theory of dynamic programming. Technical Report P-550, The RAND Corporation, 1954.

M. Ben Ahmed, F. Zeghal Mansour, and M. Haouari. A two-level optimization approach for robust aircraft routing and retiming. *Computers & Industrial Engineering*, 112:586–594, 2017. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2016.09.021.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 9780521833783.

M. Bruglieri, F. Pezzella, O. Pisacane, and S. Suraci. A variable neighborhood search branching for the electric vehicle routing problem with time windows. *Electronic Notes in Discrete Mathematics*, 47(2015):221–228, 2015.

G. Bruno, G. Improta, and A. Sgalambro. Models for the schedule optimization problem at a public transit terminal. *OR Spectrum*, 31(3):465–481, 2009.

I. Buchmann. Battery University, 2014. URL `http://batteryuniversity.com`.

S. Bunte and N. Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1: 299–317, 2009.

L. Cadarso and R. de Celis. Integrated airline planning: Robust update of scheduling and fleet balancing under demand uncertainty. *Transportation Research Part C: Emerging Technologies*, 81:227–245, 2017. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2017.06.003.

P. Carraresi, L. Girardi, and M. Nonato. Network models, lagrangean relaxation and subgradients bundle approach in crew scheduling problems. *Computer-Aided Transit Scheduling, Proceedings of the Sixth International Workshop*, pages 188–212, 1995.

V. Chiraphadhanakul and C. Barnhart. Robust flight schedules through slack re-allocation. *EURO Journal on Transportation and Logistics*, 2(4):277–306, Nov 2013. ISSN 2192-4384. doi: 10.1007/s13676-013-0028-y.

T.H. Cormen, C. Stein, R.L. Rivest, and C.E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511.

G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Column Generation*. GERAD 25th Anniversary Series. Springer, New York, 2005. ISBN 978-0-378-25485-4.

J. Desrosiers, J.B. Gauthier, and M.E. Lübbecke. Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, 236:453–460, 2014. doi: 10.1016/j.ejor.2013.12.016.

ECOpoint. Emission standards: Europe: Heavy-duty truck and bus engines, 2018. URL `http://www.dieselnet.com/standards/eu/hd.php`.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. ISSN 1097-0037. doi: 10.1002/net.20033.

R. Freling. *Models and Techniques for Integrating Vehicle and Crew Scheduling*. PhD thesis, Tinbergen Institute, Erasmus University Rotterdam, 1997.

M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, 1979.

A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study 2*, 1974:82–114, 1974.

R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 64(5):275–278, 09 1958.

K. Gwilliam, M. Kojima, and T. Johnson. Reducing air pollution from urban transport. Report 40325, The World Bank, Washington D.C., 2004.

M. van Hagen. *Waiting Experience at Train Stations*. PhD thesis, Universiteit Twente, 2011.

D. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston, 1997.

M. Huang and J-Q. Li. The shortest path problems in battery-electric vehicle dispatching with battery renewal. *Sustainability*, 8(7):607, 2016.

D. Huisman. *Integrated and Dynamic Vehicle and Crew Scheduling*. PhD thesis, Erasmus Universiteit Rotterdam, 2004.

D. Huisman, R. Jans, M. Peters, and A.P.M. Wagelmans. Combining column generation and lagrangean relaxation. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, GERAD 25th Anniversary Series, pages 247–270. Springer, New York, 2005.

M-Y. Kao, T-W. Lam, W-K. Sung, and H-F. Ting. A decomposition theorem for maximum weight bipartite matchings. *SIAM Journal on Computing*, 31(1):18–26, 2001.

N. Kliewer, B. Amberg, and B. Amberg. Multiple depot vehicle and crew scheduling with time windows for scheduled trips. *Public Transportation*, 3:213–244, 2012.

M. E. van Kooten Niekerk, J. M. van den Akker, and J. A. Hoogeveen. Scheduling electric vehicles. *Public Transport*, 9(1):155–176, Jul 2017. ISSN 1613-7159. doi: 10.1007/s12469-017-0164-0.

M.E. van Kooten Niekerk. *Cyclic Crew Scheduling*. Master's thesis, Departement Informatica, Universiteit Utrecht, 2010.

L.G. Kroon, R. Dekker, and M.J.C.M. Vromans. Cyclic railway timetabling: A stochastic optimization approach. In F. Geraets et al (Eds.), editor, *Algorithmic Methods for Railway Optimization, Revised Selected Papers, Lecture Notes in Computer Science Vol. 4359*, pages 41–66. Springer, Berlin, 2007. doi: 10.1007/978-3-540-74247-0_2.

A. Lajunen. Energy consumption and cost-benefit analysis of hybrid and electric city buses. *Transportation Research Part C*, 38:1–15, 2014.

A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

A.M. Law. *Simulation Modeling and Analysis*. McGraw Hill, fourth edition, 2007.

B. Lawson. The Electropaedia - Battery and Energy Technologies, 2014. URL `http://www.mpoweruk.com`.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. ISSN 1097-0037. doi: 10.1002/net.3230110211.

J-Q. Li. Transit bus scheduling with limited energy. *Transportation Science*, 48(4):521–539, 2014.

L. Li, H.K. Lo, and X. Cen. Optimal bus fleet management strategy for emissions reduction. *Transportation Research Part D*, 41:330–347, 2015.

L. Lindgren. Full electrification of lund city bus traffic - a simulation study. Technical report, Department of Industrial Electrical Engineering and Automation, Lund Institute of Technology, 2015.

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

R. van Nes. *Design of Multimodal Transport Networks*. PhD thesis, Technische Universiteit Delft, 2002.

L. Nurhadi, S. Borén, and H. Ny. A sensitivity analysis of total cost of ownership for electric public bus transport system in swedish medium sized cities. *Transportation Research Procedia*, 3:818–827, 2014.

N. van Oort. *Service Reliability and Urban Public Transport Design*. PhD thesis, TRAIL Research School, Delft, the Netherlands, 2011.

Oxford University Press. Oxford living dictionaries, 2018. URL `https://en.oxforddictionaries.com/definition/robustness`.

L. Peeters. *Cyclic Railway Timetable Optimization*. PhD thesis, Erasmus Universiteit Rotterdam, 2003.

A-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12:17–30, 2009.

M. Piester and M. Thorhauge. Samfundsøkonomiske fordele i køreplaner ved hjælp af passagerforsinkelsesmodeller. Master's thesis, Danmarks Tekniske Universitet, Kongens Lyngby, 2010.

M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964. doi: 10.1093/comjnl/7.2.155.

J. Reuer, N. Kliewer, and L. Wolbeck. The electric vehicle scheduling problem - a study on time-space network based and heuristic solution approaches. In *13th Conference on Advanced Systems in Public Transport CASPT 2015, Proceedings, Rotterdam*, pages 1–15, 2015.

O. Sassi and A. Oulamara. Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches. *International Journal of Production Research*, 55(2):519–535, 2017. doi: 10.1080/00207543.2016.1192695.

M. Schneider, A. Stenger, and D. Goeke. The electric vehicle routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.

G-H. Tzeng, C-W. Lin, and S. Opricovic. Multi-criteria analysis of alternative-fuel buses for public transportation. *Energy policy*, 33:1373–1383, 2005.

UNFCCC. Paris agreement, 2015. URL `http://unfccc.int/files/meetings/paris_nov_2015/application/pdf/paris_agreement_english_.pdf`.

M.J.C.M. Vromans. *Reliability of Railway Systems*. PhD thesis, Erasmus University Rotterdam, 2005.

D.P. Williamson and D.B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, 2011.

C. Yan and J. Kung. Robust aircraft routing. *Transportation Science*, 52(1):118–133, 2018. doi: 10.1287/trsc.2015.0657.

Index

Nederlandse samenvatting

Het hele proces van plannen van openbaar vervoer, van het maken van een dienstregeling tot het plannen van de voertuigen en chauffeurs, is in de afgelopen decennia uitgebreid onderzocht door wetenschappers over de gehele wereld. Dit heeft geresulteerd in een verregaande automatisering van het planningsproces. De meeste planningen en optimalisaties worden nu berekend door computers in plaats van door de planners. Dit heeft geleid tot een forse verbetering van de kwaliteit en de efficiency van het openbaar vervoer.

De methodes die op dit moment worden toegepast in programmatuur voor het plannen en optimaliseren van openbaar vervoer maken veelal nog steeds gebruik van gewoontes en aannames uit de tijd van het handmatig plannen. Ook onderzoek neemt deze gewoontes en aannames vaak over als uitgangspunt voor automatisering. De vraag is of dit niet beter kan.

Er is de laatste jaren een enorme hoeveelheid gegevens van de uitvoering van het openbaar vervoer beschikbaar gekomen. Het inbouwen van GPS-apparatuur in alle bussen heeft het mogelijk gemaakt elk voertuig op de seconde en de meter te volgen, en deze gegevens te registreren. Verder heeft de invoering van de OV-chipkaart in Nederland het mogelijk gemaakt het reisgedrag van de passagier in de bus nauwkeurig vast te leggen.

Het onderzoek zoals beschreven in dit proefschrift heeft als doel het optimaliseren van het plannen van het openbaar vervoer te verbeteren door gebruik te maken van zo veel mogelijk beschikbare data en door het kritisch onder de loep nemen van gewoontes en aannames binnen het plannen van het openbaar vervoer. Daarnaast hebben we als doel het openbaar vervoer op een duurzame en milieuvriendelijke manier te plannen.

In de eerste paar hoofdstukken van dit proefschrift zullen we de manier van plannen en optimaliseren onder de loep nemen, waarbij we aannames en gewoontes kritisch bekijken en gebruik maken van de beschikbaarheid van eerder genoemde gegevens. In de hoofdstukken daarna bespreken we uitbreidingen van de planning van het openbaar vervoer om rekening te houden met de toegenomen aandacht voor het milieu.

In Hoofdstuk 3 bespreken we het bepalen van de optimale planmatige rijtijden. De in Nederland alom gebruikte percentielen-methode is uitgebreid onderzocht in van Oort [2011]. De bij de Nederlandse Spoorwegen gebruikte methode zoals beschreven in Kroon et al. [2007] laat de percentielen-methode los en geeft betere resultaten, maar is niet zonder meer toe te passen op openbaar vervoer per bus of tram. De voornaamste reden is dat een bus of tram niet op elke halte kan wachten op de geplande vertrektijd. Wij introduceren hier enkele methoden om deze optimale planmatige rijtijden te bepalen en vergelijken deze methode met elkaar en met de percentielen-methode. Het gebruik van *continuous optimization* leidt in onze experimenten tot het beste resultaat en vermindert de gemiddelde vertraging met 60 procent.

Deze methode heeft ook als voordeel uitstekend overweg te kunnen met grote hoeveelheden data.

Een verdere vernieuwing is het loslaten van het minimaliseren van het aantal vertraagde haltepassages of de gemiddelde vertraging per halte, maar het minimaliseren van de gemiddelde reis- en wachttijd van de passagier. Alleen al deze verandering kan de gemiddelde wachttijd per passagier halveren, terwijl de gemiddelde reistijd met 2 tot 5 procent vermindert.

In Hoofdstuk 4 behandelen we de planning van de voertuigen zodat alle ritten gereden worden, de zogenaamde voertuigomlopen. In het klassieke plannen wordt voor elke rit een vaste vertrektijd en aankomsttijd gehanteerd en heeft een rit een vaste duur. Verder wordt er op basis van afspraken een minimale tijd tussen twee ritten gehanteerd om zo vertragingen op te kunnen vangen. Hedendaags onderzoek ten behoeve van het meer robuust maken van deze voertuigplanning richt zich op het juist verdelen van deze tijd tussen twee ritten, onder meer door het verschuiven van ritten. Zie hiervoor Amberg et al. [2011]. Al het onderzoek dat ons bekend is heeft gemeen dat de duur van de rit een constante waarde heeft. Als een rit bij voorbeeld 1,5 minuut te laat vertrekt, dan komt deze ook 1,5 minuut te laat aan.

Onze benadering is dat we de rijtijdmetingen en -analyses gebruiken om de werkelijke ritlengte van een rit te bepalen. We nemen niet het gemiddelde van de gemeten ritlengtes, maar beschouwen de ritduur als een kansverdeling. Tijdens het rijden van een rit komt het geregeld voor dat een voertuig te vroeg is en moet wachten tot zijn geplande vertrektijd. Dit zal minder vaak voorkomen als het voertuig met vertraging vertrekt, een vertraagd aangevangen rit duurt naar verwachting korter dan een op tijd aangevangen rit. Daarom hebben we aparte kansverdelingen voor ritlengtes gemaakt bij vertraagd vertrek. Uit de praktijk weten we dat tijdens een vertraagd aangevangen rit de tijd kan worden ingehaald, dat zien we in onze modellering terug.

Op deze manier kunnen we de invloed van twee opeenvolgende ritten op elkaar en ook op de rest van de voertuigomloop meten en berekenen. Met het zetten van goede doelen en grenswaarden hebben we de vooraf afgesproken minimale tijden tussen twee ritten niet meer nodig. Sterker nog: tot op heden is het gebruikelijk een rit niet aan te vangen voordat de vorige volgens planning is aangekomen. Ook deze restrictie laten we los.

We zien in de resultaten dat we bij gelijkblijvende kosten de punctualiteit met 2 tot 3 procent kunnen verbeteren en de gemiddelde vertraging met 30 tot 40 procent kunnen verminderen. We hebben dit vergeleken met andere methodes uit de literatuur, zoals Naumann et al. [2011], en hebben aangetoond dat onze methode beter presteert op dit vlak.

In Hoofdstuk 5 bespreken we het maken van een voertuigplanning voor elektrische voertuigen. Op dit moment is de accucapaciteit van een elektrisch voertuig nog niet zo groot dat een hele dag kan worden gereden zonder bij te laden. Tussentijds bijladen is noodzakelijk. Dit kan gebeuren op een eindhalte van een lijn, of tijdens een extra binnenkomst op de garage. Er zijn in diverse steden pilot-projecten met elektrische bussen, die allen gemeen hebben dat tussentijds bijladen niet nodig is, of in de dienstregeling makkelijk past. Bij grootschaligere planningen zal dit niet meer het geval zijn. Daarom hebben we vier methoden ontwikkeld om de eigenschappen van elektrische bussen te integreren in de voertuigplanning. De vier methodes hebben elk een andere set voorwaarden die kunnen worden meegenomen in de planning, en elk een andere nauwkeurigheid. Nauwkeurigheid en een grotere set voorwaarden gaan ten koste van de snelheid. Welk van de vier methodes toegepast kan worden hangt van

de grootte van de voertuigplanning af en met welke eigenschappen van elektrische voertuigen rekening moet worden gehouden.

In onze experimenten hebben we aangetoond dat onze methoden van voertuigplanning werken; voor de planning van omvangrijke dienstregelingen is nader onderzoek nodig om de snelheid en de kwaliteit te verbeteren.

In Hoofdstuk 6 bespreken we het plannen van bussen tijdens de overgangsfase naar schonere voertuigen. We merken op dat in de huidige literatuur hier slechts beperkt aandacht aan besteed wordt en tonen aan dat het rekening houden met een gemengd wagenpark nut heeft. In Sectie 6.3 bespreken we de uitbreiding van het plannen van voertuigen zodat met doelstellingen voor luchtvervuiling rekening wordt gehouden. We bespreken een praktijksituatie van het openbaar vervoer per bus rond Utrecht, waarin een wagenpark wordt gebruikt bestaande uit voertuigen van verschillende ouderdom en verschillende uitstootkarakteristieken. Het doel is om de totale uitstoot te minimaliserem en in het centrum van de stad Utrecht in het bijzonder. We tonen aan dat we in de praktijk zonder extra kosten de uitstoot met 15 procent kunnen verminderen en in het centrum van Utrecht zelfs met 30 procent. In Sectie 6.4 bespreken we het plannen van een gemengd wagenpark van elektrische en dieselbussen. We beschrijven een methode om een gemengd wagenpark te plannen en vergelijken deze met de aanpak waarbij een volledig elektrisch wagenpark wordt gepland en een deel van de voertuigomlopen door dieselbussen wordt gereden. We tonen aan dat onze methode om vanaf het begin voor een gemengd wagenpark te plannen leidt tot een kostenbesparing van 3 tot 4 procent ten opzichte van de volledig elektrische planning waarin dieselbussen op elektrische voertuigomlopen worden ingezet.

Curriculum Vitae

Marcel van Kooten Niekerk was born on June 7th 1974 in Haarlem, the Netherlands. In 1992 he received his VWO-diploma from the Niftarlake College in Maarssen.

From 1992 to 1996, he studied mathematics at Utrecht University. After working for an IT-company for a short while, he started his career in public transport. First as a bus driver and a tram driver at Connexxion in Nieuwegein, soon followed by a job in scheduling and optimizing the public transport. In order to improve his knowledge of optimization, he studied computer science at Utrecht University from 2007 to 2010 and successfully completed the MSc programme on Algorithmic Systems. His master's thesis was entitled 'Cyclic Crew Scheduling'. This thesis was supervised by Dr.ir. Marjan van den Akker and Dr. Han Hoogeveen. This study was completed cum laude.

In 2009, he left Connexxion and started working for Qbuzz, a new public transport company. In 2011 he started as a PhD student in the Department of Information and Computing Sciences at Utrecht University under the supervision of Dr.ir. Marjan van den Akker, Dr. Han Hoogeveen and Prof.dr. Jan van Leeuwen.