Development of a Ridematching Model

Recognizing Carpool Possibilities Based on Overlap in Routes of Users

Master Thesis

Author: J.C. Quint
May 2014

Document Title	Development of a ridematching model: recognizing carpool possibilities based on overlap in routes of users
Date	16-05-2014
Author	J.C. Quint j.c.quint@alumnus.utwente.nl
Educational Institution	University of Twente *The Netherlands*
Faculty	School of Management and Governance
Educational Program	Industrial Engineering and Management Production & Logistic Management

Graduation Committee

University of Twente	Dr. ir. M.R.K. Mes School of Management and Governance Department Industrial Engineering and Business Information Systems
	Dr. ir. J.M.J. Schutten School of Management and Governance Department Industrial Engineering and Business Information Systems
Significant	ir. K.W.J. Idzenga Managing Consultant

UNIVERSITY OF TWENTE.

significant.

PO Box 217 7500 AE Enschede +31 534 89 91 11 info@utwente.nl www.utwente.nl	Thorbeckelaan 91 3771 ED Barneveld +31 342 40 52 40 KvK 3908 1506 info@significant.nl www.significant.nl

Preface

Handing in the final version of this thesis, means the end of an inspiring, joyful, and bumpy ride, which started in Enschede in 2007, and ends in Utrecht, seven years later. Via the paved road of the University of Twente I visited the bachelor and master studies of Industrial Engineering and Management, and by now I know a thing or two about Production & Logistics. Especially due to the specialization courses of my master study, I began to know my strengths, weaknesses, and interests. With this knowledge I was able to give my ride a clear direction during the search for a master assignment.

The seven-year ride brought me many adventures, of which my time at Significant has most certainly been an important adventure. The last six to seven months of the ride, let's say 'the last mile', have been of great value to me. First, I really developed my 'professional' skills, because I got the opportunity to work and speak with inspiring, intelligent, and experienced people every day. Besides this, Significant gave me the opportunity to write my master thesis on a very interesting and evolving topic. With great interest I will follow the developments in this area the coming years. I really hope that this master thesis can provide insights that are of equal value to Significant as Significant has been to me.

During this last mile, I got support from some persons that really deserve a mentioning. First of all, I would like to thank Koen, my supervisor at Significant. Koen, it was a great pleasure working with you. Your critical assessment, mainly in the first few months, really pushed my thesis to a higher level. Somehow you motivated me to go the extra mile in my research. Most of all, I appreciate that we could both discuss work related issues and less serious topics. Furthermore, I would like to thank all other employees for their willingness to register their rides and for making me feel at home. I also thank my university supervisors, Martijn and Marco, for their critical assessment of my thesis and their clear feedback. I always left the meetings with a positive feeling and a healthy motivation to work towards the next meeting.

During a ride through crowded areas, as I would describe my ride, a lot of people flash by. However, some of the persons flashing by, stick in your mind. Normally, these are the extraordinary persons, such as the person in a monkey suit on roller-skates. Because I do not want to provide any of my friends or family with the label "person in a monkey suit on roller-skates", I will not mention these persons by name. I just want to say thanks to all my friends and family who flashed by during the ride, but still form an important part of my life. I hope to see you in future rides.

Finally, I am extremely thankful for the unlimited support and love from my parents. Knowing I can always count on your support motivates me to keep challenging myself.

Job Quint

Utrecht, 2014

Summary

Commuter traffic of Significant differs a lot from that of traditional businesses, because employees are quite flexible in where and when they work. Aiming at a reduction of costs and greenhouse gas emission of commuter traffic, Significant wants to stimulate carpooling amongst its employees. However, due to the flexibility in workplace and working times, employees do not have clear insight into their colleagues' whereabouts and therefore only accidentally share a ride. The objective of this research is to

Design a method that facilitates carpooling amongst Significant's employees
by matching colleagues for shared rides

The employees of Significant indicate that, in case of carpooling, the carpool regularly starts at a pre-arranged meeting point, such as a Park and Ride or train station. Consequently, an important requirement of a carpool facilitating method is to recognize carpool possibilities, where driver and passenger start the carpool at a meeting point chosen by the method. Further functionalities that the method should have, are to minimize detours for all users, only match users with overlap in time schedules, and take into account capacity of the car. We suggest that this method should be an offline matching algorithm and match rides on a daily basis. This means that the method should be run once a day when all rides for the next day are presumably known. To the best of our knowledge, no carpool app exists that is able to meet these requirements.

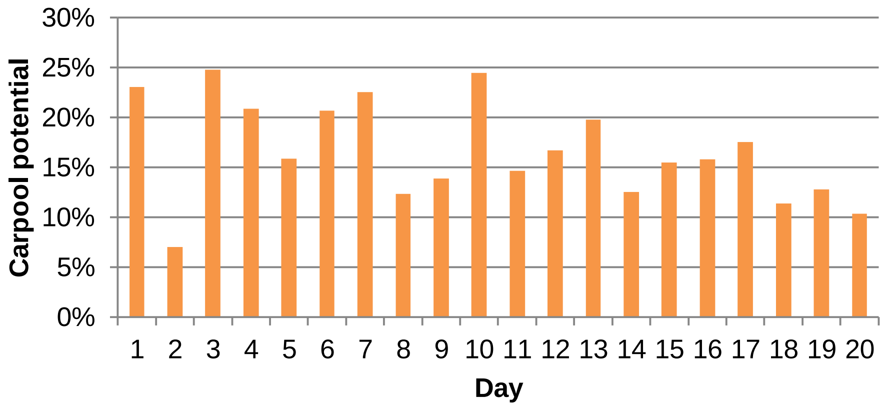
Method

Inspired by the assumptions of Ghoseiri et al. (2011), who state that when the starting point and destination of a passenger are on the driver's route, a carpool does not require a detour for the driver, we develop a method for carpooling without detours, which contradicts with prevailing methods in literature. We state that the route of each user consists of several successive points. Our method searches for two (or more) users with as much successive points in common, and then matches them for a carpool starting at the first common point. We divide this method in two components. First, shortest path computations that determine the successive points of each user. Second, a ridematching model that searches for overlap in users' routes and matches users with overlapping routes for a carpool. In our research, we assume shortest path computations are known and we therefore focus on the development of the ridematching model. We build this ridematching model in three stages (each stage is an extension of the previous stage):

- *Basic Ridematching Model*
  The model filters overlap in routes and sets up carpools between persons with overlapping routes when the passenger is able to arrive at his destination with the set of offered rides. Passenger and driver can drive to a meeting point to start the carpool, and the passenger can hop between carpools.
- *Time & Capacity Extension*
  Besides the basic functionalities, the model filters overlap in time schedules and sets up carpools between persons with overlapping time schedules when the driver is able to pick up the passenger (capacity).
- *Return Restrictions*
  Besides the basic, time, and capacity functionalities, the model sets up carpools between persons, when each person is able to reach all his destinations in a certain time period and, when the car is parked at a meeting point, is able to return to his car on a later ride. Due to the return restrictions, the model has to cope with multiple rides per person in a certain time period.

Results

To evaluate the performance of the ridematching model we introduce in this research, we collected data of Significant-related rides of all employees during four weeks. Experiments indicate that, by applying the ridematching model to the collected rides, Significant employees could have saved 7% to 25% of the distance driven each day (see figure below). On the 1416 rides we collected, 511 rides could have been (partly) saved by carpooling.



In the experiments, we observe that the running time of the model increases as the daily number of rides increases. However, for the daily number of rides of Significant personnel (50 to 90 rides), the running times (between 1 and 7 minutes) are still acceptable for an offline matching algorithm. When we compare the performance of our method with that of existing apps, we conclude that our method is able to generate carpool matches that are not recognized by existing carpool apps. For instance, in case two rides show overlap, while origins of both rides are not close to each other.

Recommendations & Further Research

For further development of the method we suggest in this research, we recommend to contact companies (such as Google, Microsoft, and TomTom) that have access to traffic data. Furthermore, we recommend to pay attention to human resistance, when introducing the use of a carpool app, and to involve potential users in the initialization of the model’s settings and parameters. For users that search for a carpool app that gives each user immediate response after entering the required data (such as starting point, destination, and time window), and allows the user to choose between different carpool offers, we recommend to use a heuristic approach that requires less computation time than our model.

Our proposed model is only tested on the set of rides collected at Significant during a four-week period. Further research should assess the model’s validity and performance on other ride sets. We also suggest to conduct further research in more reliable shortest path computations than the computations used for the experiments, extension of the model with public transport possibilities and functionalities for users not owning a car, more than one route per user per ride, the application of the model for other purposes than carpooling, and adjustments to cope with uncertainties in rides, such as the incorporation of real-time traffic information.

Table of Contents

1. Introduction

With the rising fuel prices and traffic congestion, car driving becomes less attractive in the world of economic crisis and rush we live in. However, we have to arrive at work, on our workdays, and public transport can be uncomfortable in peak hours. Carpooling, simply travelling with multiple persons in one car, is considered as an interesting alternative (Morency, 2007), since it saves fuel costs and, when applied on a large scale, reduces traffic congestion. Furthermore, sharing rides reduces the emission of greenhouse gasses (Massaro et al., 2009).

When entering 'carpool' in the Google search engine, the first search results are carpool meeting places nearby and the other search results are tools/apps for matching drivers (offering a ride) with passengers (seeking a ride). These results typify the developments of carpooling in recent years. Dutch government and municipalities have put a lot of effort into creating and improving carpool meeting places. Due to devices such as smart phones, connected to GPS and social networks, we are able to quickly find or offer a ride. Because of this, the number of websites and apps for finding a travelling companion is growing considerably. This research builds on these 'carpool apps' and suggests new functionalities for matching drivers and passengers.

This chapter introduces the research by a short description of the company under consideration in Section 1.1, followed by the motivation for the research in Section 1.2. Section 1.3 introduces the problem where our research focuses on. Section 1.4 presents the boundaries of the research and hereafter we state the research goal in Section 1.5. Finally, we state research questions and our research approach in Section 1.6 and Section 1.7.

1.1 Company Description

Significant was founded in 2003 as a consultancy and research firm. Its head office is located in Barneveld, the Netherlands. Operating in the Dutch public sector, Significant conducts research at and advices the government, provinces, and municipalities, but also educational institutions, healthcare institutions, healthcare insurers, and healthcare providers. Currently, the company employs 53 employees of which 42 consultants/researchers. Significant is competent in quantitative and qualitative research as well as consultancy in organizational development, business administration, and purchasing. The firm can be divided in five working areas, namely:

- Healthcare
- Legislation
- Mobility
- Social Security
- Business Administration

This research is part of the Mobility area, which mainly focuses on consulting for the government in large and complex projects concerning public transport and mobility issues. Examples of activities in this area are supporting a municipality in the public tendering of bus transport and the development of a model to calculate (social) costs and benefits of bus fleet formations.

1.2 Motivation

Since the business area Mobility is continuously supporting authorities in organizing public transport and mobility in urban and rural areas, it is familiar with subjects such as reducing traffic congestion and the carbon footprint. From this perspective, the Mobility group started to think about the commuter traffic of the Significant employees.

The consultants of Significant are scattered across the country, working at the clients' offices. Approximately four out of five workdays are spent outside the office in Barneveld. It is possible that two or more consultants work at the same location simultaneously. This location can be a city or area, but more specific, a clients' office or the office of Significant in Barneveld. Furthermore, the carpooling opportunities increase when broadening to a pool of multiple companies. For example, Karzoo, a website that connects carpoolers, suggests "to merge the journeys of personnel with those of neighbouring companies". Since the office of Significant is located at a business park with two other companies, it is possible to form a pool of the employees of these three companies to facilitate ride-sharing. Another possibility might be to share rides with clients, in the case of employees working at clients' offices. Although it seems there are plenty of opportunities for ride-sharing, carpooling is a rare phenomenon at Significant.

Aiming at a reduction of costs and greenhouse gas emission of commuter traffic, Significant wants to research the possibilities to reduce its number of single-person trips by car. An additional advantage is that the travel time of the employees can be spent effectively in case of carpooling. This means that colleagues can discuss work-related issues during the ride. Things normally discussed during office hours, can be shifted to a ride home, ride to work, or a ride to the client. This saves time at the office, enabling people to be more productive. A shared ride can also simply be used to integrate with colleagues and discuss non work-related subjects.

1.3 Problem Description

Commuter traffic of Significant differs a lot from that of traditional businesses. With traditional businesses we mean companies with a fixed workplace, such as factories and shops. At these traditional businesses, employees have to go to the same location every day. At Significant, employees are quite flexible in where and when they work. They can have meetings across the country, work at clients, work at the office of Significant in Barneveld, or work at home. Consequently, an employee does not know when which colleague is working at the same place, which impedes carpooling. It is not feasible for employees to call or email all colleagues to find out which colleagues are heading the same direction. So currently people only incidentally carpool, for instance in case of a joint meeting.

Employees' flexibility in workplace and working times makes carpooling at firms such as Significant more complex, compared to carpooling at traditional businesses. However, Significant expects a reduction in the number of single-person trips by car, when employees are more aware of their colleagues' whereabouts. An increase in shared rides amongst employees leads to a reduction in the costs of commuter traffic as well as a reduction in CO2 emission. Therefore, we define the following problem statement.

Significant's employees lack insight into their colleagues' whereabouts and therefore only accidentally share a ride to/from the office in Barneveld or to/from a client. This leads to unnecessarily high costs and CO2 emission of Significant's commuter traffic.

1.4 Research Scope

For a successful carpool match between driver and passenger(s), two primary preconditions must be met (Handke & Jonuschat, 2013):

- Route convergence. Carpool possibilities arise when driver and passenger share (partly) the same route. The driver can pick up and drop off passengers en route. However, in practice perfect route convergence rarely exists. So it is important to determine detour boundaries that drivers and passengers still accept.
- Time convergence. Besides route constraints, carpooling is subject to time constraints. Travel times of driver and passenger have to coincide to make a shared ride possible. However, similar to the route convergence, also time convergence can encompass some tolerance. The use of earliest and latest departure times, as well as earliest and latest arrival times, for both driver and passenger, enables tolerance in time constraints. The driver can pick up the passenger as long as their departure time from the pick up point is larger than both their earliest departure times, and smaller than both their latest departure times.

So for matching two (or more) carpool participants, it is important to realise that their routes as well as their travel times converge.

One of the main barriers to carpooling is sharing a ride with strangers (Correia & Viegas, 2011). In this research, we assume that the travellers are no strangers of each other, but members of a closed user group, such as colleagues or members of the same sports club. Following this assumption, we assume that the travellers are willing to carpool. So no further barriers to carpooling exist, except an unclear view on the carpool possibilities amongst acquaintances.

In contrast to Barbucha (2013), who states that ride-sharing is complementary to public transport, and Gruebele (2008), who assumes that a traveller can hop between different transportation modes, such as train, car, and bus, we ignore the possibility of travelling with any form of public transport. This research would become too complex when taking into account public transport restrictions concerning route and time schedules. In our research, we do take into account the possibilities for a traveller to drive to a carpool meeting point in his own car, or to hop between different carpools.

1.5 Research Goal

In Section 1.3, we described that it is difficult for Significant employees to undertake a carpool, because people lack insight into their colleagues' whereabouts. In order to facilitate the carpooling process, we therefore define the research goal:

Design a method that facilitates carpooling amongst Significant's employees
by matching colleagues for shared rides

Employees will not have to be pro-active in searching for colleagues heading the same direction anymore, when the method makes the match. This way we remove the barrier for employees of lacking insight in colleagues' whereabouts.

In matching employees, the method should take into account constraints concerning delays in travel time (compared to the situation without carpooling), detours and returns of passengers (every picked-up passenger also needs to get back). We elaborate on the constraints of the method in Chapter 4. Inputs for the method, such as travel times, starting points, and destinations of trips should be derived from (digital) employees' agendas.

The method aims at facilitating ride-sharing amongst employees of Significant; however, it should use a widely applicable methodology. This means the method should be able of facilitating ride-sharing for any closed user group (a group of acquaintances), with comparable characteristics.

1.6 Research Questions

In Section 1.5, we stated that this research aims at developing a supportive carpool method. To divide this objective into manageable pieces, we define six research questions.

I. *What is the current situation?*

   a. *Regarding commuter traffic at Significant?*
      We gather information on current rides driven by Significant personnel. This information consists of starting points, destinations and times of rides driven. With these data, we are able to estimate the potential of carpooling at Significant.

   b. *Regarding carpooling at Significant?*
      We describe how employees currently undertake a carpool.

   c. *Regarding carpool apps?*
      Answering this question provides information on how current carpool tools work and the pros and cons of existing tools.

   The answers to these sub questions expose the current conditions at Significant and at the carpool app market.

II. *What are the key characteristics a carpool app should have in order to facilitate Significant's commuter traffic?*
   With the knowledge obtained from sub questions I.a and I.b, we define which functionalities should be included in a carpool app in order to be of assistance to the employees. We compare these functionalities with existing apps derived from sub question I.c. This should give us a starting point for possible solutions.

III. *Which methods exist in literature to facilitate ride-sharing?*
   After answering sub question II, we know how the functionalities of a desired app relate to that of existing apps. A literature study must provide methods that enable the creation of an alternative to existing tools. We search for methods in the area of ride-sharing literature.

IV. *Which assumptions do we make for developing a carpool method with approaches from the literature?*
   We discuss which assumptions we make to combine insights gained from literature with own views on carpool facilitation, in order to develop a carpool method. Besides this, we elaborate on the objectives of such a method, and on the constraints that the method should take into account.

V. *Which method is suitable to facilitate carpooling?*
   We present a method for facilitating carpooling amongst Significant's employees. As stated in Section 1.5, this method should also be applicable to any closed user group with similar characteristics to that of Significant.

VI. *Which benefits can be realized with the developed carpooling method?*
   We use the rides driven by Significant personnel (gathered for answering question I.a) to evaluate the performance of the carpooling method. This indicates whether the deployment of a carpool app using the suggested method, is useful for Significant and comparable groups.

Figure 1-1 shows an overview of the research questions and corresponding chapters.

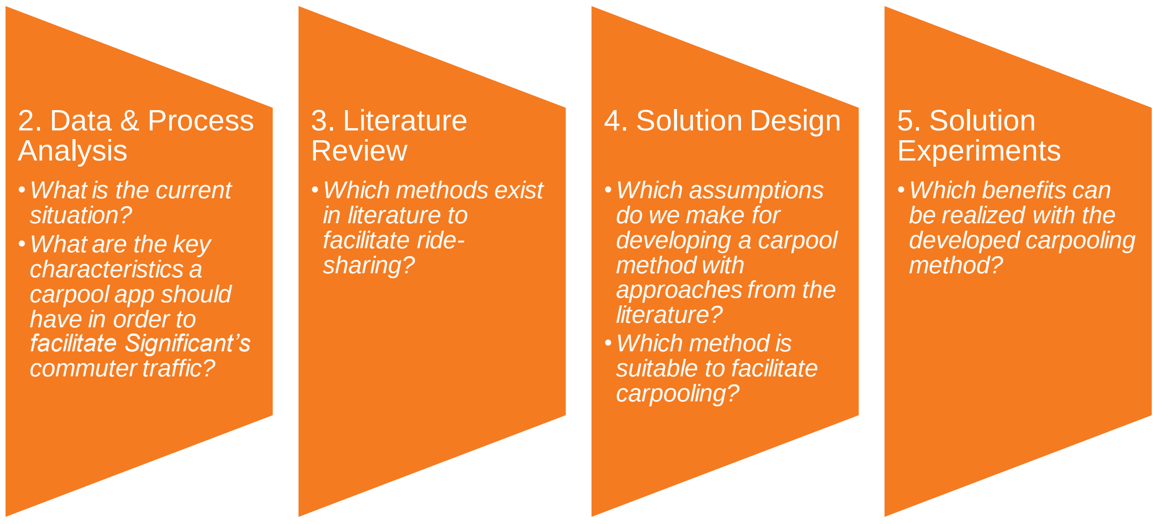

2. Data & Process Analysis
• What is the current situation?
• What are the key characteristics a carpool app should have in order to facilitate Significant's commuter traffic?
3. Literature Review
• Which methods exist in literature to facilitate ride-sharing?
4. Solution Design
• Which assumptions do we make for developing a carpool method with approaches from the literature?
• Which method is suitable to facilitate carpooling?
5. Solution Experiments
• Which benefits can be realized with the developed carpooling method?

Figure 1-1: An overview of the forthcoming chapters. The chapters give an answer to the accompanying research question(s).

1.7 Research Approach

The outline of this research is as follows. We start with a data and process analysis in Chapter 2. In that chapter, we analyse the commuter traffic of Significant, by collecting rides driven by employees during a given period. Besides this, we examine how employees currently arrange a carpool, and we describe the functionalities of currently available carpool apps. In Chapter 3, we perform a literature review, in which we search for methods that are able to match persons for a carpool. We design the method that should facilitate carpooling amongst Significant's employees in Chapter 4. Subsequently, we conduct experiments on this method in Chapter 5. Chapter 6 provides the conclusions and limitations of this research. Furthermore, we make recommendations for implementing our method in a real life carpool app, and suggest areas for further research.

2. Data & Process Analysis

In this chapter, we answer two research questions.

- *What is the current situation?* We divide this research question into three subjects. In Section 2.1, we elaborate on the commuter traffic of Significant. For this, we collect data of Significant's commuter traffic for a 4-week period. Section 2.2 describes how carpooling currently is arranged at Significant and Section 2.3 discusses existing carpool apps.
- *What are the key characteristics a carpool app should have in order to facilitate Significant's commuter traffic?* Section 2.4 answers this second research question. We elaborate on the functionalities of a future carpool app suitable for Significant and how these functionalities relate to that of current carpool apps.

Finally, we conclude this chapter in Section 2.5.

2.1 Commuter traffic Significant

In order to be able to draw some conclusions concerning the commuter traffic of Significant, we collect data of Significant-related trips of all employees during four weeks. In this section, we elaborate on these data.

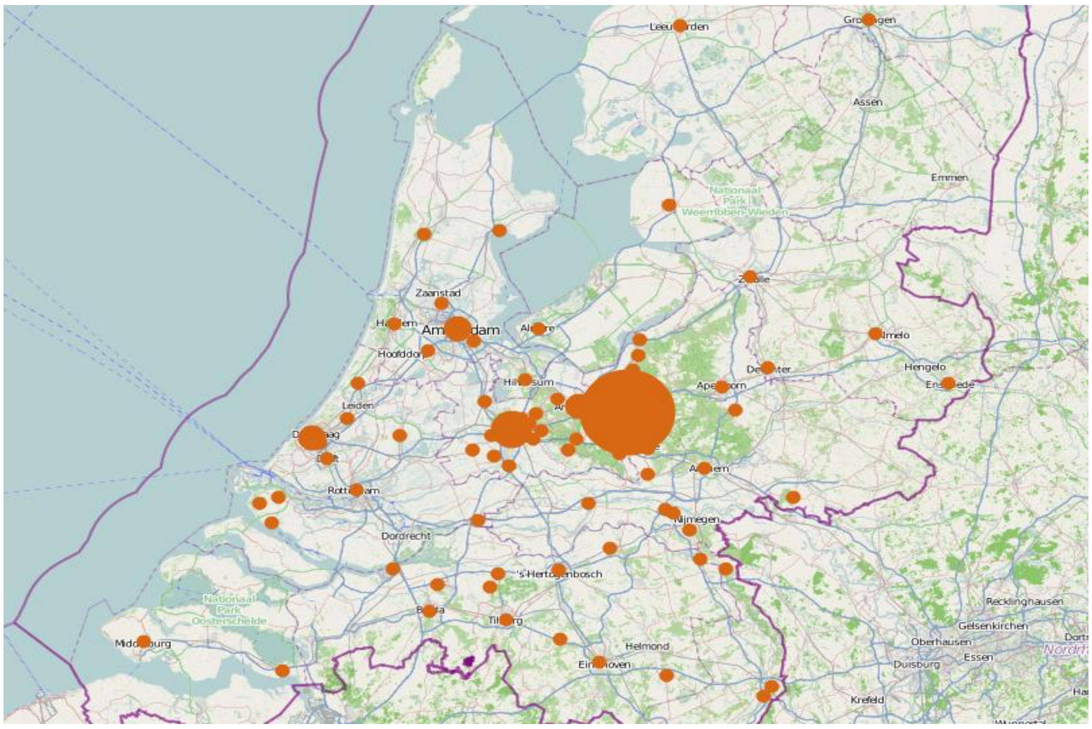
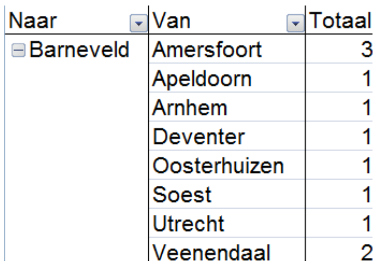


Figure 2-1: Destinations of observed trips. The size of the circle indicates the visiting frequency of the location. The biggest circle corresponds with the office of Significant in Barneveld, which means that this office is visited the most of all locations.

In total, we collected 1416 one-way trips in 4 weeks. In Figure 2-1, we map the destinations of these trips. Most of the trips finished at the office of Significant in Barneveld. Other popular destinations are Utrecht, Amsterdam, and The Hague. The most driven trajectory (starting point – destination) is Barneveld – Utrecht. In general, the data indicate that mainly trips to and from the office in Barneveld are potentially suitable for carpooling. Consider for example Table 2-1. In total, 11 trips are made towards Barneveld between 8 and 9 am on a Monday. Three of these trips started in Amersfoort and two of the trips started in Veenendaal. Instead of five individual trips, two joint trips could have been made, under the assumption that all passengers, leaving their car at home or at a parking place, have been able to meet their travel needs for the remainder of that day. Then, also the trips from Oosterhuizen, Apeldoorn, and Deventer could be combined, because a large part of these trips are similar. The same applies to the trips from Arnhem and Veenendaal (see Figure 2-2).



Naar	Van	Totaal
Barneveld	Amersfoort	3
	Apeldoorn	1
	Arnhem	1
	Deventer	1
	Oosterhuizen	1
	Soest	1
	Utrecht	1
	Veenendaal	2

Table 2-1: Trips to Barneveld on a Monday morning between 8 and 9 am.

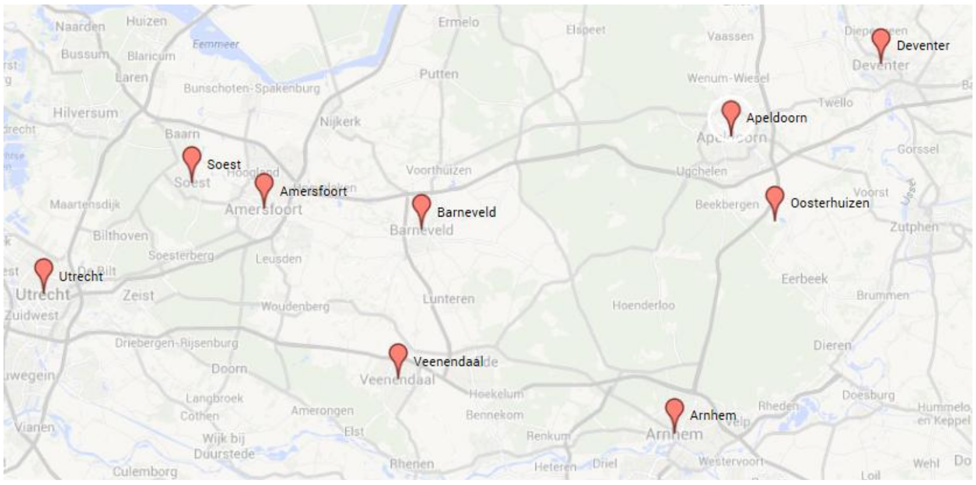


Figure 2-2: The locations of Table 2-1 plotted on a map (Google).

Besides the rides made by Significant employees, we also kept track of the carpools during this 4-week period. In total, we observed 7 carpools, which means that 0.49% of all trips (1416) were shared rides.

2.2 Carpooling at Significant

In this section, we describe how a carpool currently is arranged at Significant. This should form a basis for the identification of key characteristics of a carpool app suitable for Significant, which we discuss in Section 2.4.

In the problem description (Section 1.3), we already briefly described how a carpool is currently arranged at Significant. Employees only incidentally carpool, for example when having a joint meeting. To describe the current process of arranging a carpool, we analyze the carpools from our data collection (Section 2.1). In total, we recognized 7 shared rides in 4 weeks time. One of these shared rides was from the Significant's office to a drink amongst colleagues and five of the rides were to or from a meeting at the office. This supports the statement that employees of Significant mostly share a ride when having a joint meeting.

From conversations with employees, we know that in case of carpooling, the driver and passenger do not necessarily live close together. When having a joint meeting at a customer who is located in the outskirts of the

Netherlands, employees sometimes meet at a Park and Ride or train station and undertake a carpool from that point on. Starting the carpool at a pre-arranged meeting point is eased by the fact that every employee of Significant has a company car. The ability for all participants to drive to the meeting point, enables carpooling from places that are difficult to access for pedestrians or public transport, such as parking lots along highways. A carpool is arranged by e-mail or by a telephone call.

So the characteristics of carpooling at Significant are:

- Only in case of a joint meeting
- Sometimes start at a meeting point
- Arranged by e-mail or telephone call

2.3 Current Carpool Apps

This section describes the functionalities of currently available carpool apps. We discuss two carpool apps in particular, namely an app available for the employees of a hospital (Section 2.3.1) and an online available app (Section 2.3.2). We conclude this section with an overview of currently available carpool apps, which we base on the functionalities of the apps from Sections 2.3.1 and 2.3.2.

2.3.1 Carpool App ZGT

In this section we take a closer look at a carpool app that is currently used for the commuter traffic of a hospital. By analyzing this carpool app, we get a clear view of how current carpool tools operate.

In 2013, Ziekenhuisgroep Twente, further referred to as ZGT, developed a carpool app in cooperation with Twente Mobiel and a software developer (see text box at the right side of the page for more information on ZGT and Twente Mobiel). With this app, the ZGT employees are able to offer or seek rides. For ZGT, the main goals of the app are to reduce occupied parking lots at ZGT hospitals and to create bonding amongst colleagues. Secondary goals are to reduce traffic congestion in the cities of Twente and to reduce the emission of greenhouse gasses by ZGT commuter traffic. Of course, the latter goals are important for Twente Mobiel.

ZGT is a hospital group located in Twente, consisting of 2 hospitals and 5 outpatient clinics.

Twente Mobiel is a collaboration between different organizations in Twente, aiming at helping employers with programs concerning mobility for their employees, in order to keep Twente accessible, livable, and clean.

Figure 2-3 gives a schematic overview of the possible actions a user of the ZGT carpool app can take. As the figure shows, the users first have to create an account before they can use the ZGT carpool app. In this step, the user provides basic information, such as ZGT e-mail, phone number, and home address. The ZGT e-mail is used for verifying whether the user is a ZGT employee, because the app is only meant for ZGT employees.

After creating an account, the user is directed to the homepage (see Appendix A). This homepage is shown every time the user logs in and gives an overview of the rides the user offered, the rides for which the user subscribed (to drive along), passenger requests that the user still has to accept/reject, and carpooling history. From this homepage, the user can go to the page to modify his account, the ride offer page, or the ride search page. In Appendix A, we describe the page where rides can be offered and the page where rides can be searched in detail by showing and describing screenshots of the app. In this section, we elaborate on the criteria used by the app to assess whether a ride seeker matches with a certain offered ride.

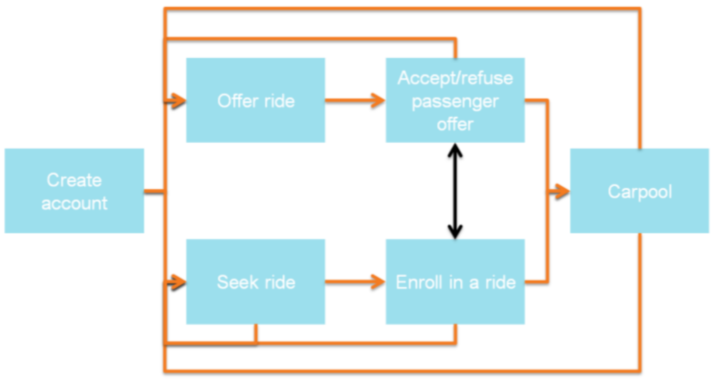


Figure 2-3: Possible steps for a user of the ZGT carpool app.

Based on some criteria, a matching algorithm determines which rides are presented as carpool options to the ride seeker. We show the criteria of the matching algorithm of the ZGT carpool app in Figure 2-4. Remarkably, the app does not take 'time' (more specific than the date) into account when matching ride seekers with offered rides. In the remarks at the end of this section, we come back to the absence of this criterion. Criteria that the matching algorithm does take into account are:

- Trip criterion. A match only exists if the searched ride and offered ride have a starting point (ZGT location) or destination (ZGT location) in common.
- Date criterion. Offered rides and searched rides with the same date are matched.
- Radius criterion. This criterion is only considered when the ride searcher has entered a maximum radius, which is optional.

Suppose we search a ride from work to our home. After entering the required data, the app shows all available rides that start at the same ZGT location, at the same day. These rides meet the trip criterion and the date criterion (orange criteria in Figure 2-4). When entering data for searching a ride, we can optionally set a maximum radius. This adds another criterion to the matching algorithm (blue criterion in Figure 2-4). The app now only shows the offered rides with a destination within the maximum radius of our home address.

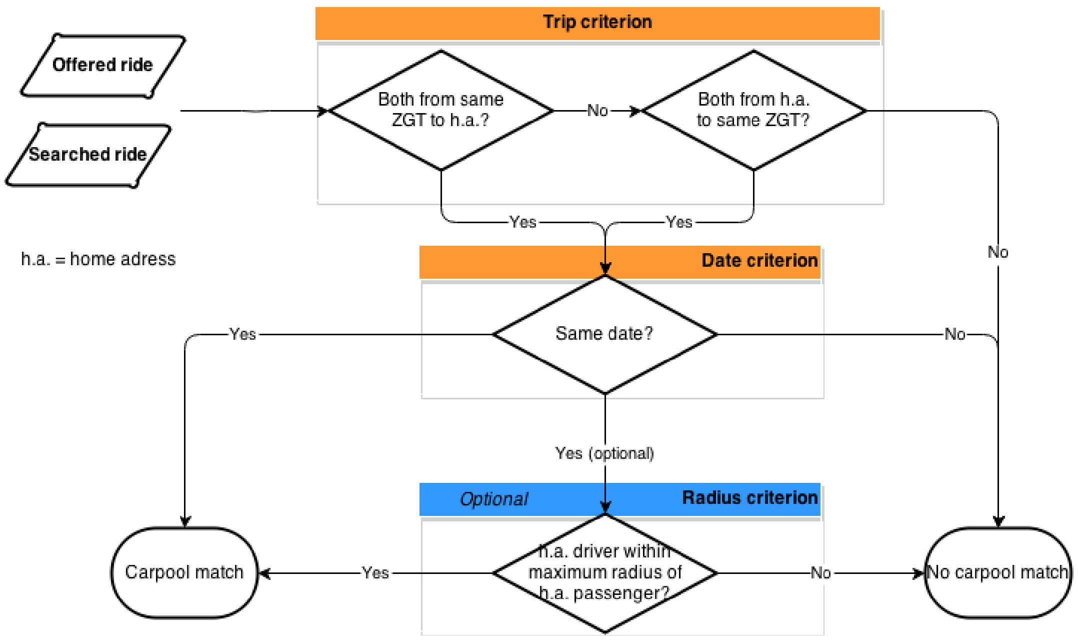


Figure 2-4: Criteria of the matching algorithm of the ZGT carpool app.

After analyzing the ZGT carpool app, some remarks can be made.

- By giving the possibility to set a maximum radius, the app excludes possible carpool matches. One employee's house can be located on a colleague's route from home to a ZGT location. This certainly is a carpool possibility, however when the colleague's house is not located within the maximum radius, the app excludes this possibility.
- Because the app purely focuses on commuter traffic of ZGT, the user can only offer or search for rides from or to a ZGT location. It is not possible to offer or search for rides that do not include a ZGT location.
- As we mention earlier in this section, the app does not take into account 'time'. However 'time' is one of the two primary preconditions for a carpool (see Section 1.4). When no overlap in time windows exists, then there are no carpool opportunities. Not incorporating a 'time' criterion means that the ZGT carpool app suggests carpools to ride seekers that could be non-feasible carpool opportunities. When searching for a ride in the morning, also the rides of that day's afternoon or evening are shown.
- The main reason for the previous remark is that the ride searcher is made aware of ride offers outside his time window, but meeting his starting point/ destination criteria. This triggers a ride searcher to frequently use the carpool app.
- The ZGT carpool app does not take into account return trips. The user has to use the app again for the return trip. So first a ride from home to work has to be searched with the app and for a trip from work to home, a new ride has to be searched with the app.

In the following section, we continue with the analysis of another carpool app. This carpool app functions different from the app of ZGT that we discussed in this section.

2.3.2 Carpool App Flinc

Section 2.3.1 established some drawbacks in the carpool app of ZGT, including:

- The app excludes potentially feasible carpool matches by the possibility to set a maximum radius.
- By not taking 'time' into account, the app suggests infeasible carpools to ride seekers.

An existing carpool app that overcomes these drawbacks is Flinc, an online ride-sharing network that "matches carpools automatically and in real-time via PC, App and integrated into navigation systems" (Flinc). Flinc is an 'open app', anyone can create an account. The users can choose whether they would like to be matched with direct contacts (social network contacts), members of a subgroup (companies, universities, etcetera), or with anyone in the Flinc network.

Similar to the ZGT app, a user first has to create an account and subsequently arrives at the homepage. The homepage of Flinc also provides the user with an overview and the possibility to navigate towards either the ride offer or ride search page. In this section, we discuss the criteria Flinc presumably uses to match ride searchers with offered rides. In Appendix B, we explain how we derived these criteria from the carpool app.

Figure 2-5 shows the probable matching process of Flinc. In contrast to the ZGT carpool app, searched and offered rides do not necessarily have similar starting points or destinations for a match in the Flinc carpool app. A searched and offered ride can have different starting points as well as different destinations, and yet lead to a match. Flinc determines the match between an offered and a searched ride, based on the detour the driver has to make to pick up and drop off a passenger (lower criterion in Figure 2-5). When a driver has to make a detour larger than presumably 40 kilometres (see Appendix B), then Flinc does not suggest a carpool possibility. Another criterion, used by the carpool app of Flinc, concerns time constraints. The time criterion checks whether the arrival time of an offered ride at the starting point of a searched ride, lies within the time window of the searched ride.

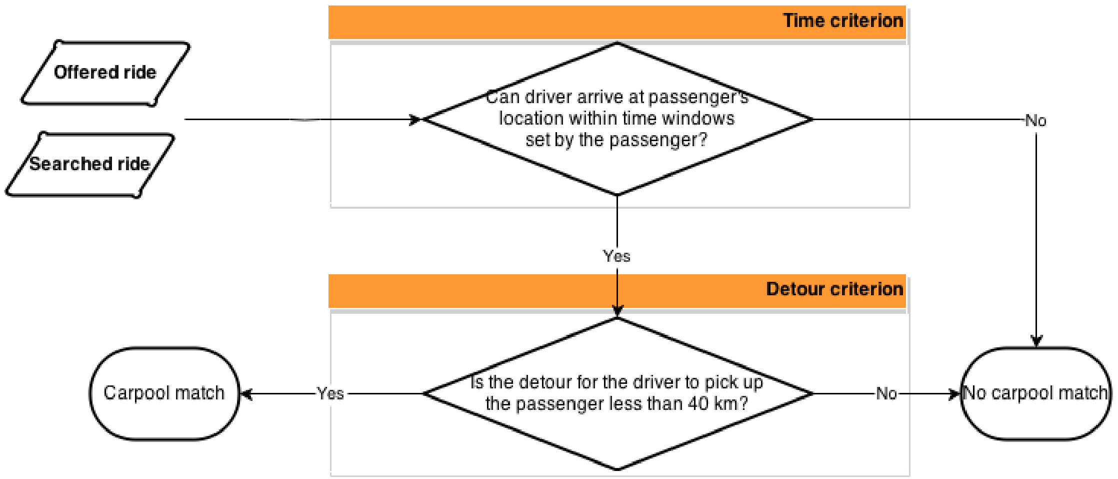

Time criterion
Offered ride
Searched ride
Can driver arrive at passenger's location within time windows set by the passenger?
No
Yes
Detour criterion
Carpool match
Yes
Is the detour for the driver to pick up the passenger less than 40 km?
No
No carpool match

Figure 2-5: Presumable criteria of the matching algorithm of Flinc. Flinc does not impose restrictions on starting point or destination, so the rides offered and searched at Flinc can depart from, and arrive at, any location.

In the beginning of this section, we already mentioned that two drawbacks of the ZGT carpool app are not applicable to Flinc. By checking whether the arrival time of an offered ride at the starting point of a searched ride fits within the time window of this searched ride, Flinc takes into account 'time'. Therefore, Flinc does not suggest infeasible carpools. Furthermore, the Flinc carpool app does not pair rides based on starting point/destination, and does not have problems caused by the possibility to set a maximum radius. When the starting point and destination of a searched ride are on the route of an offered ride, the Flinc app always generates a match, because the searched ride does not lead to a detour. However, also the Flinc carpool app excludes potentially feasible carpool matches, as we show in Figure 2-6.

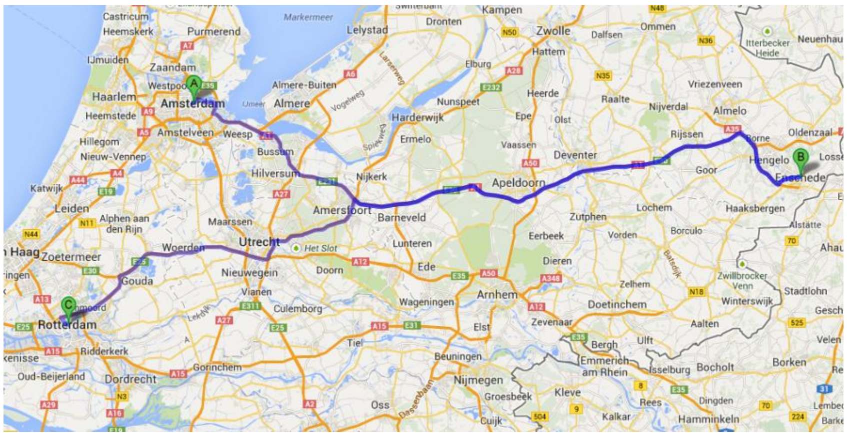


Figure 2-6: A ride from Amsterdam to Enschede and a ride from Rotterdam to Enschede, come together at a certain point, which is Amersfoort (Google). Flinc does not recognize the possibility to meet at Amersfoort.

When we search a ride from Rotterdam to Enschede, we are not matched with an offered ride from Amsterdam to Enschede. Although the corresponding routes merge at Amersfoort (see Figure 2-6), Flinc does not recognize a carpool possibility, because it assumes that the driver has to pick up the passenger in Rotterdam (which is a larger detour than 40 kilometres, see Appendix B). So a drawback of the Flinc app is the inability to cope with a variable meeting point. When the passenger travels to Amersfoort, a perfect carpool opportunity arises with the ride from Amsterdam to Enschede.

Summarizing, the carpool app of Flinc matches drivers and riders when the driver can pick up the passenger within the time windows set by the passenger and when the passenger's pickup point is on the route of the driver, or requires a slight detour (less than 40 kilometres). Major drawback of the Flinc app is the inability to cope with variable meeting points.

2.3.3 Overview

In Sections 2.3.1 and 2.3.2, we discussed the functionalities of two existing carpool apps. Based on these apps, we provide a brief overview of the current carpool app market in this section.

Our findings of the ZGT carpool app are supported by the research of Geisberger et al. (2009) who state that "today's ride sharing services still mimic a better billboard. They list the offers and allow to search for the source and target city, sometimes enriched with radial search". Corresponding to this statement, many 'billboard' apps can be found on the Internet. Examples of apps comparable to the one of ZGT are BlaBlaCar, meerijden.nu, roadsharing.com, carpoolworld.com, ridejoy.com, and carpooling.co.uk. As discussed in Section 2.3.1, drawbacks of these 'billboard apps' are:

- The app excludes potentially feasible carpool options when the user is able to set a maximum radius.
- By not taking 'time' into account, the app suggests infeasible carpools to ride seekers.

In Section 2.3.2, we discussed a carpool app with different functionalities compared to the 'billboard' apps. The Flinc app matches drivers and riders based on overlap in time windows and a detour for the driver. We assume the matching algorithm of Flinc is quite unique, because we cannot find a carpool app with the same functionalities. Due to the unique functionalities, Flinc is able to avoid some drawbacks of the 'billboard' apps. However, we also establish a major drawback in the Flinc app. Similar to the ZGT app, Flinc is not able to cope with variable meeting points. To the best of our knowledge, no carpool apps exist that can recognize carpool possibilities where driver and passenger start the carpool at a certain meeting point chosen by the carpool app.

2.4 Desired Carpooling

In the future, Significant wants to make use of a carpool app, but without the drawbacks of the apps mentioned in Section 2.1. In this section, we define which functionalities should be included in a carpool app in order to assist the employees. Besides a user ID (requiring a one-time account setup, including basic information, such as e-mail, telephone, home address, and car capacity), the user submits his starting points and destinations, and dates and times of his rides for say a week. This might be done by synchronizing with an agenda. Subsequently, the user gets a message with a suggestion for a shared ride. This message should include information on the meeting point, possible drop-off point, the date and time of the ride, and persons (including contact details) who to share a ride with.

As Figure 2-7 shows, a matching algorithm should turn the user's input into a carpooling advice (output). This matching algorithm is the method we aim for in the research goal (Section 1.5). Karp (1992) makes a distinction between online and offline algorithms. "An online algorithm receives a sequence of requests and must respond to each request as soon as it is received. An offline algorithm may wait until all requests have been received before determining its responses” (Karp et al., 1990). The apps discussed in Section 2.3 run their ridematching algorithm every time a ride is entered in the system and thus make use of an online algorithm. The immediate response to the user is a main advantage of an online system, but problems can occur when newly entered rides enable better carpool opportunities than suggested to the user. Besides this, when all requests are directly served, the possibility arises that new requests enter an ‘empty’ system. In this case, running a matching algorithm makes no sense. As mentioned earlier in this section, the required input might be translated from the user’s agenda. Therefore, a user probably does not expect immediate response. In contrast to current carpool apps, we suggest to use an offline algorithm and run it once a day when all rides for the next day are presumably known, for example at 6 pm.

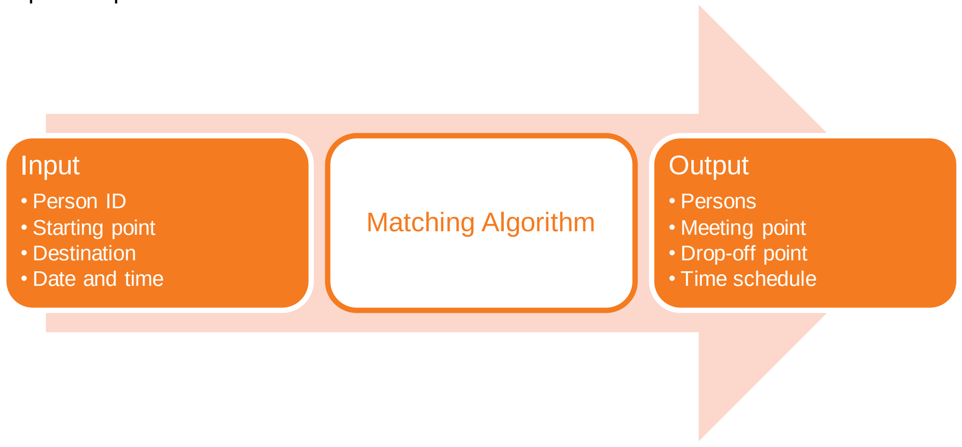


Figure 2-7: Overview of the desired app.

Compared to the existing apps (Section 2.3), the input to the desired app is the same. However, some differences in the output of the apps can be noticed. The output of the ZGT carpool app consists of a list of rides including starting point, destination, and times. The desired app of Significant should generate a concrete carpooling advice or advices including participants, meeting point, possible drop-off point, and time schedules (when to be at which place?) of the carpool.

In Section 2.1 we already remarked that the ZGT carpool app excludes carpool opportunities by the possibility to set a maximum radius. Furthermore it does not take into account time windows and matches drivers and riders only on starting point or destination and day of the ride. The Flinc app has different functionalities, but also lacks the ability to cope with variable meeting points. Significant wants to overcome these shortcomings by using another matching algorithm. We list the characteristics of the matching algorithm of the desired carpool app:

- Minimize detours. With time as a scarce resource and employee resistance concerning delays, Significant requires an app that provides the user with (a list of) solutions within certain detour boundaries.
- Variable meeting point. As described in Section 2.2, the employees occasionally use meeting points when carpooling. A carpool app should also encompass carpools where driver and passenger meet at a parking lot, a Park and Ride, etcetera. The user does not have to specify the meeting point in advance. This meeting point is rather an output of the app (see Figure 2-7).
- Time windows. The app should match travellers based on their time schedules, so matched drivers and riders must have comparable arrival/departure times at the meeting and drop-off points.
- Offline matching algorithm. Only once all requests are entered in the system, drivers and passengers should be matched. We suggest to run the matching algorithm once a day when all rides for the next day are presumably known, for example at 6 pm.

- Capacity. Another characteristic of the desired matching algorithm, which is also used in the ZGT app as well as in the Flinc app, is the inclusion of multiple persons per car. Each driver should be able to indicate how many passengers he can give a ride.

In accordance with the conclusion in Section 2.3.3 that no carpool apps exist that can recognize carpool possibilities where driver and passenger start the carpool at a certain meeting point, we conclude that, to the best of our knowledge, no carpool app exists that is able to meet all described functionalities of the desired app of Significant.

2.5 Conclusion

In this chapter, we discussed the current situation at Significant concerning their commuter traffic and carpool arrangement, furthermore we elaborated on current carpool apps. Based on this knowledge, we described which functionalities a future carpool app of Significant should have.

We collected data on Significant-related trips (1416 one-way trips) of all employees during a 4-week period. In general, the data indicate that mainly trips to and from the office in Barneveld are potentially suitable for carpooling. The identified characteristics of current carpooling at Significant are:

- Only in case of a joint meeting
- Sometimes start at a meeting point, such as a Park and Ride or train stations
- Arranged by e-mail or telephone call

The current carpool app market can be divided in two classes:

- 'Billboard' apps. This category of carpool apps generate a list of offers based on source and target cities, and sometimes encompass a radial search.
- Flinc app. The Flinc app matches drivers and riders based on overlap in time windows and detour for the driver. As far as we know, the Flinc app is a unique tool on the carpool app market.

However, to the best of our knowledge, no existing app is able to recognize carpool possibilities where driver and passenger meet at a meeting point chosen by the carpool app. Precisely this functionality is one of the characteristics of the desired carpool app of Significant. Further functionalities of Significant's desired app are:

- Minimize detours. The app should provide the user with (a list of) solutions within certain detour boundaries.
- Time windows. The app should match travellers based on their time schedules.
- Offline matching algorithm. Only once all requests are entered in the system, drivers and passengers should be matched.
- Capacity. Each driver should be able to indicate how much passengers he can give a ride.

3. Literature Review

This chapter provides an answer to the following research question.

- *Which methods exist in literature to facilitate ride-sharing?*

First, Section 3.1 provides a general overview of carpooling and the concepts developed over time. In Section 3.2, we introduce methods to match drivers and riders. In these methods, routing plays a major role, so in Section 3.3 we discuss methods to determine the shortest route. Finally, in Section 3.4, we give a summary of this literature review.

3.1 Carpooling

In literature, no consensus exists on the terms ‘carpooling’ and ‘ride-sharing’. Evans et al. (1985) indicate that ride-sharing is “the shared use of a transportation vehicle by more than one person”, including for example carpools, vanpools, and public transport services. Others say that ride-sharing only includes carpools and vanpools (Brownstone & Golob, 1992). The common view is that ride-sharing consists of different approaches, including carpooling (Rosenbloom, 1978). However, most authors interchangeably use ‘carpooling’ and ‘ride-sharing’. For instance, Chaube et al. (2010) state that “ride-sharing or carpooling is a privately organized shared use of a passenger vehicle for commuting purposes”. Jacobson & King (2009) and Vanoutrive et al. (2012) also consider carpooling and ride-sharing as the same concept. In this thesis, we join the latter group and therefore assume that the concepts carpooling and ride-sharing are identical.

According to Chan & Shaheen (2012), the development of carpooling over time can be divided into five key phases:

- **World War II.** During the Second World War, US government promoted ride-sharing amongst habitants, because of oil and rubber shortages. This led to an increase in shared rides, however after the war the focus on carpooling weakened (Ferguson, 1997).
- **Energy crisis.** The carpooling concept returned during the oil crisis in the seventies. As a response on the limited availability of oil resources, the number of shared rides increased tremendously (Bonsall, 1981).
- **Early organized ride-sharing schemes.** During the late eighties and early nineties, the goal of carpooling shifted from energy conservation to reducing air pollution and traffic congestion, but never reached high levels, because of low gasoline prices (Chan & Shaheen, 2012). Transportation agencies made employers responsible for trip reduction of their employees. This way employer-organized ride-sharing schemes arose (Dill & Wardell, 2007).
- **Reliable ride-sharing systems.** Around the year 2000, the Internet became a widely used phenomenon. Although carpooling still suffered from a dip, the Internet enabled online ridematching platforms (Chan & Shaheen, 2012).
- **Technology-enabled ridematching.** This phase builds on ride-sharing concepts developed in earlier phases, but integrated with latest technologies and developments, such as social networks and mobile devices with Internet and GPS. With these latest technologies, ridematching services are able to match either non-recurring or recurring trips on a very short notice or up to several days in advance (Deakin et al., 2010).

Throughout these phases several carpool concepts were invented. In the light of the desired situation at Significant, we zoom in on some concepts:

- **Flexible carpooling**. Simply carpooling without pre-arrangement. Riders gather at a fixed meeting point and wait in a queue for a ride, comparable to a taxi stand (Minett & Pearce, 2011).
    - Informal flexible carpooling (also known as casual carpooling or slug lines) is an open flexible carpooling system, which sets no boundaries at the participation in the pool (Minett & Pearce, 2011). Thus, everyone can go to the pickup point and take or offer a ride.
    - Formal flexible carpooling is a closed flexible carpooling system. After pre-screening, persons can become a member of such a pool. For all members, a participation record is kept (Minett & Pearce, 2008).
- **Dynamic ride-sharing** (also known as real-time ride-sharing (Amey et al., 2011)). Ride-sharing enabled by "a system where an automated process provided by a ride-share provider matches up drivers and riders on very short notice or even en-route" (Agatz et al., 2010). This concept is developed due to the rise of Internet-enabled mobile devices.
- **Multi-hop ride-sharing.** Ride-sharing that takes into account the possibility for a rider to transfer between drivers (Drews & Luxen, 2013). By hopping between drivers, riders can reach destinations that go far beyond the destination of one driver.

We outlined particularly these three concepts, because they all relate to some extent to the desired carpooling situation of Significant, as discussed in Section 2.4. Flexible carpooling is relevant concerning the desired situation, because this concept takes into account the possibility of starting the carpool at a certain meeting point. Although the desired carpool app of Significant runs offline, dynamic ride-sharing is an interesting topic, because matching drivers and riders en route requires powerful speed-up techniques, which can be useful for an offline app too. Finally, multi-hop ride-sharing is applicable to the desired situation for recognizing the possibility to switch between cars. So for the desired situation we are looking for a mix of the concepts flexible carpooling, dynamic ride-sharing, and multi-hop ride-sharing. In Section 3.2 we elaborate on the methods concerning matching drivers and riders for this mixed concept.

3.2 Carpool matching methods

Several authors have examined the problem of matching drivers and riders and proposed a solution to this problem. As described in Section 2.4, the desired carpool app for Significant should at least meet the following conditions, when matching drivers and riders.

- The matching method should minimize detours for driver as well as passenger.
- In matching travellers, the app should take into account variable meeting points.
- The app should be able to cope with time windows.

Especially the first and third conditions are the centre of attention in carpooling literature. Herbawi & Weber (2012) call it "the ridematching problem with time windows (RMPTW)". The genetic algorithm they propose is able to solve this problem, but it assumes that pickup points (meeting points) are known in advance. Agatz et al. (2010) consider a comparable matching problem, only they assume that a driver can take just one rider. They argue that a ridematching problem becomes difficult to solve when riders can hop between drivers or drivers can pick up multiple riders (see Table 3-1).

	Single Rider	**Multiple Riders**
Single Driver	Matching of pairs of drivers and riders: *Easy*	Routing of drivers to pickup and deliver riders: *Difficult*
Multiple Drivers	Routing of riders to transfer between drivers: *Difficult*	Routing of riders and drivers: *Difficult*

Table 3-1: Rideshare variants (Agatz et al., 2010).

This matrix of Agatz et al. (2010) is endorsed by Kleiner et al. (2011) who state that assigning multiple passengers to a single driver (upper right of Table 3-1) "necessitates the deployment of heuristic algorithms", because of computational complexity.

Concerning Significant, the last row of Table 3-1 is noteworthy, because it copes with transfer points. These transfer points are important in the search for a suitable method to apply at Significant, because they enable the driver and passenger to start the carpool at a certain meeting point, instead of starting the carpool at the passenger's location. Although recognizing the complexity, some authors propose algorithms to solve the transfer-allowed (or multi-hop) ride-sharing problem. Coltin & Veloso (2013) introduce three algorithms to plan for transfer-allowed ride-sharing, but they do not elaborate on the proposed methods. On the other hand, Drews & Luxen (2013) and Herbawi & Weber (2011) present well-explained algorithms for this problem. We further discuss the methods of Drews & Luxen (2013) and Herbawi & Weber (2011).

Both the method of Drews & Luxen (2013) and the method of Herbawi & Weber (2011) assume a fixed number of stations k. Each station $s_i \in S = \{s_1, \ldots, s_k\}$ represents a possible starting point or destination for a driver or rider as well as a possible point for a rider to switch cars. Both methods also deploy the technique of time-expanded graphs (Pyrga et al., 2004) for modelling driver's offers. In a time-expanded graph "every node corresponds to a specific time event (departure or arrival) at a station and edges between nodes represent either elementary connections between the two events, or waiting within a station" (Pyrga et al., 2004). So each station that is visited by one or more drivers has at least one arrival event and one departure event (assuming a driver is eventually leaving the station). Events are sorted in chronological order. Figure 3-1 shows an example of such a time-expanded graph, with stations s_1 and s_2. The grey nodes represent departure events and are connected to the arrival events (white nodes). When a departure event is connected to an arrival event at the same station, it represents waiting within a station, and when connected to an arrival event at another station, it shows a trip. For example, the connection between t_6 and t_{10} stands for a ride offer o from station 1 to station 2, with a duration of $t_{10} - t_6$, while $c(o)$ and $d(o)$ respectively represent the costs and driver of ride offer o.

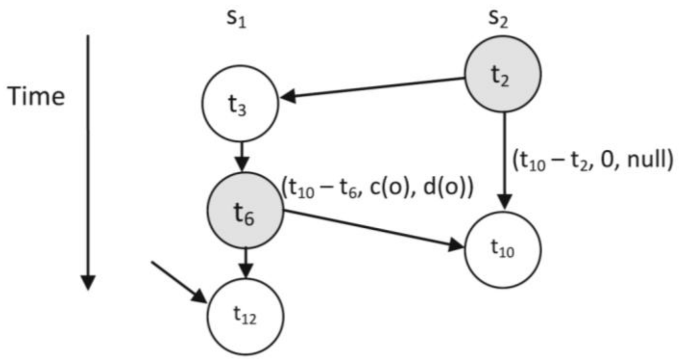


Figure 3-1: Time-expanded graph. White nodes are arrival events, gray nodes are departure events (Herbawi & Weber, 2011).

The idea of Drews & Luxen (2013) is to plot all 'potentially reasonable' nodes and arcs based on the ride offers. This means that all stations that can be visited within certain boundaries, based on a set of rides, are included in the time-expanded graph. Again we consider a ride offer o, but now alternatively defined as $o = (s, t, \tau, \delta, \varepsilon)$, with starting point s, destination t, departure time τ, and upper bounds for detour δ and delay ε. Drews & Luxen (2013) create two supersets (with $\mu(a, b)$ as distance between a and b):

$$S_1 := \{s' \in S^s \mid \mu(s, s') + \mu(s', t) \leq (1 + \delta) * \mu(s, t)\}$$

This set S_1 contains all possible starting stations s' (of subset $S^s \subseteq S$, denoting all source stations of ride seekers) that can be visited by ride o, without violating its detour constraints.

$$S_2 := \{t' \in S^t \mid \mu(s, t') + \mu(t', t) \leq (1 + \delta) * \mu(s, t)\}$$

This set S_2 contains all possible destination stations t' (of subset $S^t \subseteq S$, denoting all target stations of ride seekers) that can be visited by ride o, without violating its detour constraints. After creating the subsets, the method examines all possible combinations from $s' \epsilon S_1$ to $t' \epsilon S_2$. For example, checking which route from s' to t' satisfies detour constraints for an offered ride o is formulated as:

$$\{s' \in S_1, t' \in S_2 \mid \mu(s, s') + \mu(s', t') + \mu(t', t) \leq (1 + \delta) * \mu(s, t)\}$$

Each combination (s', t') that can be adopted in ride o (so that its path becomes (s, s', t', t), without violating delay and detour constraints), gets an arc $a_i \in A = \{a_1, \dots, a_k\}$ in the time-expanded graph with $a_i = (u_i, v_i, o_i)$, where u_i is the node that represents the departure event on station s' and v_i is the arrival event on station t'. Instead of assuming exact arrival and departure times, Drews & Luxen (2013) consider timeslots, containing waiting times at stations and buffer for unexpected delays. So $u_i = (s', \left\lceil \frac{\tau}{s_l} \right\rceil)$, which means u_i is a node on station s' in timeslot $\frac{\tau}{s_l}$ (latest departure time divided by length of timeslot). See Figure 3-2.

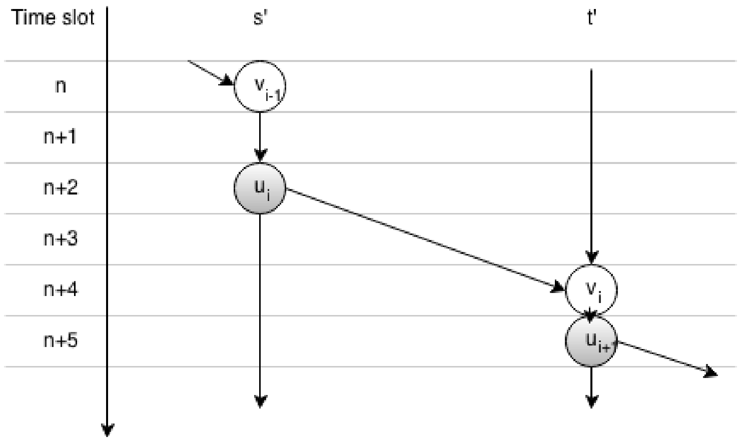


Figure 3-2: Slotted time-expanded graph. For example, the departure event at station s' occurs in time slot $n + 2$. So it takes place between $(n + 1) * (timeslot\ length)$ and $(n + 2) * (timeslot\ length)$.

Drews & Luxen (2013) opt for constructing a table for all pair wise distances between the stations using the algorithm of Knopp et al. (2007). So for determining the supersets or feasibility of paths (s, s', t', t), the algorithm just has to read the distances (and travel times) from a pre-determined table. While adding multiple ride offers to the slotted time-expanded graph, Drews & Luxen (2013) use Dijkstra's algorithm (Dijkstra, 1959) on the slotted time-expanded graph for computing a best fit ("a fit that minimizes the rider's delay") for a given ride request.

Besides algorithms for multi-hop ride-sharing, also an algorithm for multimodal ride-sharing is found in literature. In the research scope (Section 1.4), we stated that we exclude the possibility of travelling via public transport. However, methods concerning multimodal ride-sharing do include transfer points. Because the use of variable transfer points leads to useful insights, we describe the multimodal ride-sharing method of Bit-Monnot et al. (2013) in this literature review.

Similar to Drews & Luxen (2013), Bit-Monnot et al. (2013) also suggest to use pre-processing techniques to facilitate shortest path queries. Bit-Monnot et al. (2013) try to solve what they call the "2 Synchronization Points Shortest Path Problem (2SPSPP)". The passenger and driver meet at a meeting point (first synchronization point), share a ride towards a drop-off point (second synchronization point), and from there on individually continue their tours. They assume that the passenger can walk or use public transport to reach the meeting point or final destination. Figure 3-3 provides an overview of the solution approach to this 2SPSPP. In this figure, o_p and o_c are the origins of respectively the passenger and driver, x_{up} are possible meeting points, x_{off} are possible drop-off points and d_p and D_c are destinations of respectively passenger and driver. Where algorithms A_1, A_2 and A_4 solve a one-to-all shortest path problem, so algorithm A_1 calculates the shortest path from o_p to each possible meeting point x_{up}. The result of algorithms A_1, A_2 and A_4 is a set of meeting points x_{up} (meeting driver and passenger constraints) and a set drop-off points x_{off} (meeting driver constraints). The remaining algorithms (A_3 and A_5) are dedicated to solving the best origin problem ("best origin to minimize the cost at the destinations"). So given o_p and o_c, algorithm A_3 determines the best meeting point x_{up} (origin) for each potential drop-off point x_{off}. Subsequently, given D_c and the results of A_3, algorithm A_5 computes the drop-off point that leads to the lowest costs at d_p. Because this last algorithm (A_5) only minimizes costs (arrival time/ length of trip) at the destination of the pedestrian, the main goal of Bit-Monnot et al. (2013) is to optimize the passenger's trip.

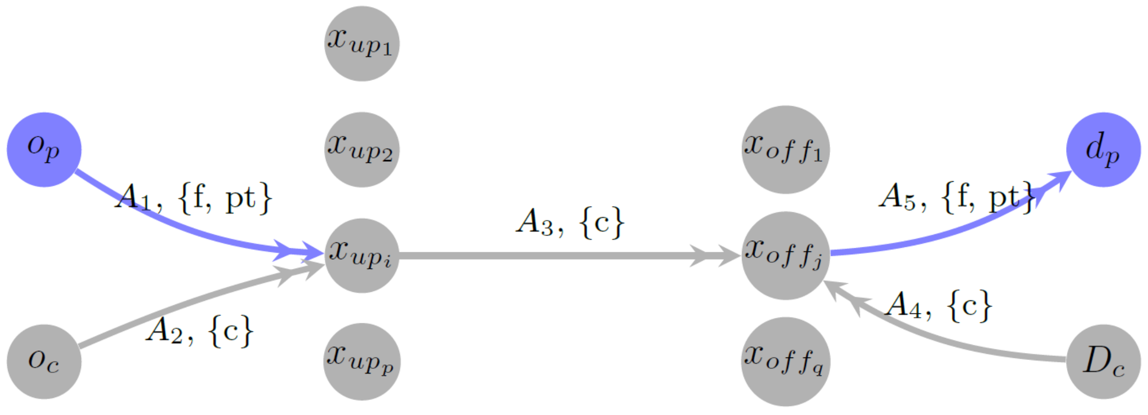


Figure 3-3: Overview of solution approach to 2SPSPP (Bit-Monnot et al., 2013).

The described methods of Drews & Luxen (2013) and Bit-Monnot et al. (2013) both assume a pre-defined set of possible meeting points and lead to the optimal solution for the passenger, within boundaries set by the driver. Unless maximum detour and delay of the driver are set to 0 (which is highly unlikely, because Bit-Monnot et al. (2013) state that the driver "is willing to take a detour"), these methods confirm statements such as "carpooling requires additional travel time to pick up and drop-off passengers" (Giuliano et al., 1990) and "carpooling requires rerouting" (Bellemans et al., 2012). While these assumptions dominate literature, Ghoseiri et al. (2011) suggest the contrary. They state that the route of driver j consists of the successive points $v_1^j, v_2^j, v_3^j, \dots, v_n^j$ and the pickup and drop-off points of rider i are v_O^i and v_D^i. Then if both v_O^i and v_D^i are on $v_1^j, v_2^j, v_3^j, \dots, v_n^j$ in the right order (so first v_O^i and then v_D^i), neither the driver nor the rider has to take a detour. Carpooling, in this case, is only possible when the entire passenger's route overlaps with (a part of) the driver's route. However, also Ghoseiri et al. (2011) introduce a detour factor in their calculations. To the best of our knowledge, no method in ride-sharing literature can be found that recognizes overlap in routes between driver and passenger.

3.3 Shortest Path Calculations

In this chapter, we already mentioned the term 'shortest path' and the methods of Knopp et al. (2007) and Dijkstra (1959). In this section, we elaborate on the shortest path problem and the corresponding solution methods.

"In graph theory, the shortest path problem is the problem of finding a path between two vertices such that the sum of the weights of the corresponding edges is minimized. It has applications in various fields like transportation, communication, routing and scheduling" (Elizabeth & Sujatha, 2012). In our case, these weights represent distance or time; the shortest path is the route that takes the minimum amount of kilometres or time.

Below, we show the mathematical formulation of the shortest path problem (Bazaraa et al., 2011). We consider a directed graph $G = (V, E)$, with a set V of m vertices and a set E of n edges. Each edge (i, j) is associated with a cost c_{ij}.

$$Minimize \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}\, x_{ij}$$

$$subject\ to$$

$$\sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} 1\ if\ i = 1 \\ 0\ if\ i \neq 1\ and\ i \neq m \\ -1\ if\ i = m \end{cases}$$

$$x_{ij} = 0\ or\ 1 \qquad\qquad i, j = 1, \dots, m$$

The objective is to minimize the costs of the path, when travelling from node 1 to node m. The constraints ensure that the shortest path is a linked path between 1 and m, and that the variable x_{ij} is either 0 or 1 (with $x_{ij} = 1$ indicating edge (i, j) is part of the shortest path and $x_{ij} = 0$ otherwise).

According to Pijls & Post (2009) the best-known methods for solving the shortest path problem are the Dijkstra (Dijkstra, 1959), Bellman-Ford (Bellman, 1956; Ford & Fulkerson, 2010), and A* (Hart et al., 1968) algorithms. In contrast to Dijkstra's algorithm, the Bellman-Ford algorithm can solve the shortest path problem on graphs with negative weights (Skiena, 2008). This does not apply to this thesis' subject, because no negative distances exist on a road network. The A* algorithm is similar to Dijkstra's algorithm, except A* uses estimates to speed up the algorithm (Beker et al., 2011). In Appendix C, we show how Dijkstra's algorithm works.

The network we use in Appendix C to calculate the shortest path with Dijkstra's algorithm contains 14 nodes. In the calculation we settle 12 nodes for finding the shortest path, which means we review distances between the origin node and 12 nodes of the network. On average, in a network of n nodes, n/2 nodes have to be settled to find the shortest path (Delling et al., 2009). According to Sanders & Schultes (2005), the real road network of Western Europe consists of approximately 18 million nodes, meaning that the average search space (number of settled nodes) is 9 million nodes in case of using the basic Dijkstra algorithm for finding a shortest path on this network. This huge search space makes the algorithm far too slow to be applied to such datasets (Bauer et al., 2010), so route planning systems make use of modified versions of Dijkstra's algorithm (Wagner & Willhalm, 2003). Because both Dijkstra's algorithm and the A* search start at the origin point and keep running until the search reaches the destination (Pijls & Post, 2009), Pohl (1971) suggests a bi-directional search for Dijkstra's algorithm and A*, to achieve computational savings. The bi-directional search simultaneously starts at the origin as well as the destination, and stops when the forward search (starting at the origin) and backward search (starting at the destination) meet 'in the middle' (Pohl, 1971). See Figure 3-4 for an overview of the search spaces of Dijkstra's algorithm, bi-directional search and A*.

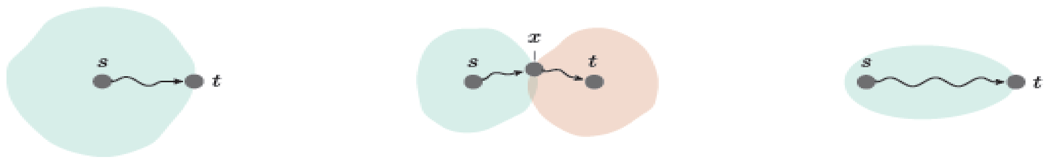


Figure 3-4: Illustration of the search spaces of Dijkstra's algorithm (left), a bi-directional search (middle) and A* (right) (Bast et al., 2014).

While A* and bi-directional search speed-up Dijkstra's algorithm with a factor of around 2, which is still impractical in case of real road networks (Abraham et al., 2013), two-stage algorithms yield speed-up factors with larger orders of magnitude (Knopp et al., 2007). In the first stage, data are pre-processed, for example by removing irrelevant nodes, laying shortcuts between nodes (pre-computing distances), or adding labels and values to nodes and edges (Abraham et al., 2013). These pre-processed data are used to accelerate shortest path calculations (second phase). In the past decade, many two-stage methods are developed, such as highway hierarchies (Sanders & Schultes, 2005), contraction hierarchies (Geisberger et al., 2008), A* with landmarks (Goldberg & Harrelson, 2005), and transit node routing (Bast et al., 2007). These methods can be categorized in two approaches (Bauer et al., 2010):

- **Hierarchical approach.** Techniques that "prune a Dijkstra search by relaxing only edges (u, v) to 'sufficiently important' nodes v" (Bauer et al., 2010).
  - *Highway hierarchies, contraction hierarchies, transit node routing*
- **Goal-directed approach.** Techniques that "direct the search toward the target t by preferring edges that shorten the distance to t and by excluding edges that cannot possibly belong to a shortest path to t" (Bauer et al., 2010).
  - *A* with landmarks*

In addition to the aforementioned speed-up techniques, also combinations exist between them, for example highway hierarchies combined with landmark-based A* or bi-directional search combined with A* (Bauer et al., 2010). Figure 3-5 shows an overview of speed-up techniques for Dijkstra's algorithm.

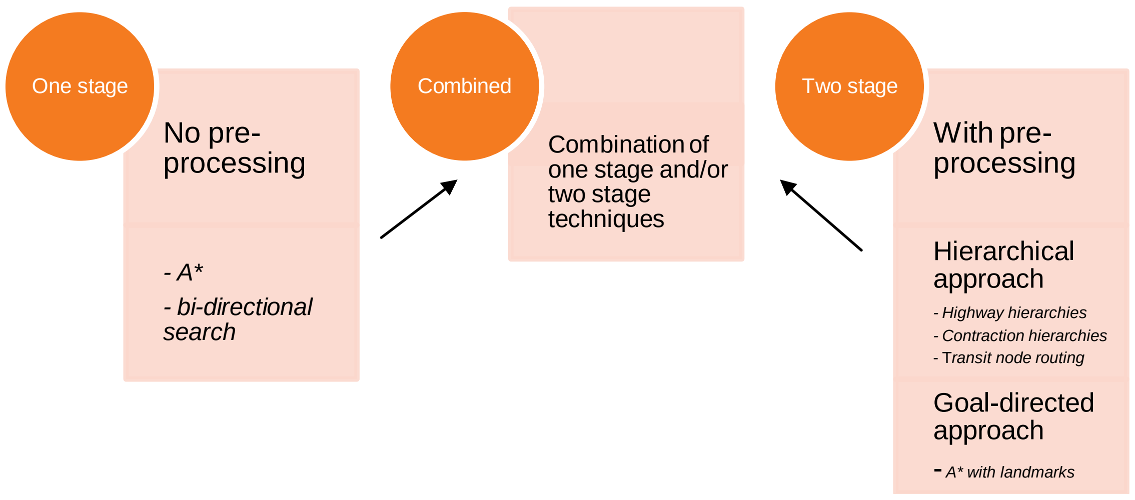


Figure 3-5: Categories of existing methods to speed-up Dijkstra's algorithm.

As mentioned earlier, many two-stage approaches are developed in the past decade. However, no single-best algorithm can be pointed out so far, because of the trade-off between query time, pre-processing time and space usage (Delling et al., 2009). Single stage methods do not require any pre-processing time, or space consumption (except the graph itself), but require a large query time. On the contrary, pre-computing distances between all nodes on the graph requires large pre-processing times and space consumption, but no query times (Akiba et al., 2013).

3.4 Conclusion

We searched the literature for methods applicable to a mixed concept of flexible carpooling, dynamic ride-sharing, multi-hop ride-sharing and multimodal carpooling. Although some authors argue that a ridematching problem becomes difficult to solve when riders can hop between drivers or drivers can pick up multiple riders, we did find authors who propose methods for solving such problems. Drews & Luxen (2013) cope with multi-hop ride-sharing by making use of a time-expanded graph. All stations that a set of drivers can visit within certain boundaries, such as maximum detour and maximum delay, get a node on this time-expanded graph. Subsequently, Drews & Luxen (2013) use Dijkstra's algorithm on the time-expanded graph for computing a best fit ("a fit that minimizes the rider's delay") for a given ride request. Dijkstra's algorithm is a method to solve a shortest path problem, as discussed in Section 3.3.

Besides algorithms for multi-hop ride-sharing, also an algorithm for multimodal ride-sharing is found in literature. To determine the meeting point and drop-off point of a carpool Bit-Monnot et al. (2013) solve a 2 Synchronization Points Shortest Path Problem (2SPSPP). Algorithms determine a set of possible meeting points (meeting driver and passenger constraints) and a set of possible drop-off points (meeting driver constraints). Another algorithm determines the best meeting point for each possible drop-off point, and a last algorithm determines the drop-off point that optimizes the passenger's trip.

The methods discussed in Section 3.2 consider a set of known meeting points and drop-off points. Algorithms determine which of these meeting points and drop-off points can be visited by a certain driver, without violating the driver's detour and delay constraints. These methods make a clear distinction between driver and passenger, and focus on optimizing the passenger's route, therefore we call them passenger focussed methods. Existing methods assume ride-sharing leads to detours for the driver. To the best of our knowledge, no method in ride-sharing literature can be found that excludes detours, by recognizing overlap in routes between driver and passenger.

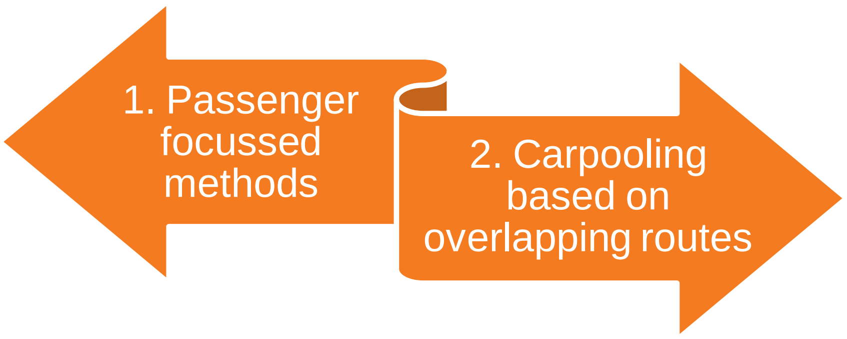

1. Passenger focussed methods
2. Carpooling based on overlapping routes

Figure 3-6: The antithesis between two possible solution methods resulting from literature research.

4. Solution Design

In this chapter, we discuss the following research questions.

- *Which assumptions do we make for developing a carpool method with approaches from the literature?*
- *Which method is suitable to facilitate carpooling?*

In Section 4.1, we discuss the ideas where our method is based on, and in Section 4.2 we elaborate on the assumptions we make for our method. Section 4.3 we present the ridematching model and elaborate on the model's functionalities. Section 4.4 compares our method with methods from literature, and Section 4.5 presents the entire ridematching model. We conclude this chapter with a short summary in Section 4.6.

4.1 Foundation

In Section 3.2, we described the methods of Drews & Luxen (2013) and Bit-Monnot et al. (2013), who optimize the passenger's route and assume a detour for the driver. However, when looking at the characteristics of the desired situation at Significant, there is an alternative way for matching drivers and riders. The alternative method is inspired by the assumptions of Ghoseiri et al. (2011), who state that the route of driver j consists of the successive points $v_1^j, v_2^j, v_3^j, \dots, v_n^j$ and the pickup and drop-off points of rider i are v_O^i and v_D^i. If both v_O^i and v_D^i are on $v_1^j, v_2^j, v_3^j, \dots, v_n^j$ in the right order (so first v_O^i and then v_D^i), neither the driver nor the rider has to take a detour. This idea combined with the characteristics of the desired situation that an app should minimize detours and everyone at Significant has a company car, leads us to the following method.

- Every user j submits his starting point v_S^j and destination v_D^j, such that a shortest path with successive points $v_S^j, v_1^j, v_2^j, \dots, v_D^j$ is planned.
- Then search for overlap in routes for users i and j, which occurs if there exists a sequence for which holds $v_g^i, v_{g+1}^i, \dots, v_{g+k}^i = v_h^j, v_{h+1}^j, \dots, v_{h+k}^j$, for $k > 0$. So search for g and h, such that k is as large as possible.
- If overlap ($v_g^i, v_{g+1}^i, \dots, v_{g+k}^i = v_h^j, v_{h+1}^j, \dots, v_{h+k}^j$ with $k > 0$) for certain users (i and j) exists, then save the first overlapping point $v_g^i = v_h^j$ as a possible meeting point for a carpool.

In contrast to prevailing methods in literature, our method is designed for carpooling without detours. We suggest that persons with actual overlapping routes form carpools. This kind of carpooling requires minimum effort of the participants, because they do not have to deviate from their original routes. So, where existing carpool apps enable the driver to take a detour in order to pick up a passenger (who has no car), but fail to recognize (partial) overlap in routes, we suggest a method that stick the driver and passenger (who has a car) to their routes and is able to recognize where their routes overlap.

Figure 4-1 gives an impression of users with overlapping routes. The paths of users i and j converge at node 3 and subsequently diverge at node 4. We state that edge (3,4) offers a pure carpool opportunity, because users i and j pass edge (3,4) anyway, and therefore enables carpooling without detours. Of course, carpooling from 3 to 4 alone does not bring both users at their destination. Because one of the users has to leave his car at node 3 in case of carpooling, node 4 should enable this user to reach his destination by public transport or by transferring to another carpool. As we already mentioned in Section 1.4, we do not take into account public transport, so in this situation, node 4 should enable the user to ride with another driver to his destination. Besides making sure that each user reaches his destination, the method also has to ensure that, on the return route, the rider returns to the place where he left his car (node 3).

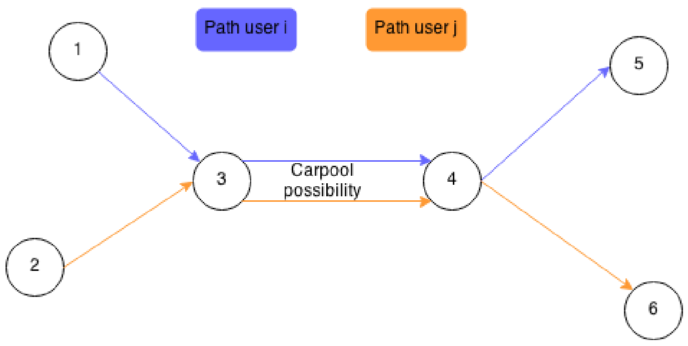


Figure 4-1: Example of two users with (partly) overlapping routes.

As described in Section 3.2, Drews & Luxen (2013) and Bit-Monnot et al. (2013) assume carpooling necessitates detours by the driver. Besides this, they only consider situations where the passenger is a pedestrian and cannot drive to a certain meeting point. As far as we know, no research has been conducted in overlapping paths or ride-sharing amongst car owners. In the following sections, we construct a method that is able to recognize overlapping routes and to suggest meeting points to start a carpool.

4.2 Model Assumptions and Input

In the description of the desired situation (Section 2.4), we mentioned a 'matching algorithm', which requires user inputs person ID, starting point, destination, and date and time of a ride. We assume a person ID encompasses basic information such as e-mail, telephone, home address, but also car capacity, as stated in Section 2.4. With the inputs, the matching algorithm matches rides and returns the persons that can form a carpool, together with meeting points, drop-off points, and time schedules. We divide this matching algorithm in two components. The first component is assigned to shortest path computations. The model that we suggested in Section 4.1, which we call the ridematching model, forms the second part of the matching algorithm, see Figure 4-2.

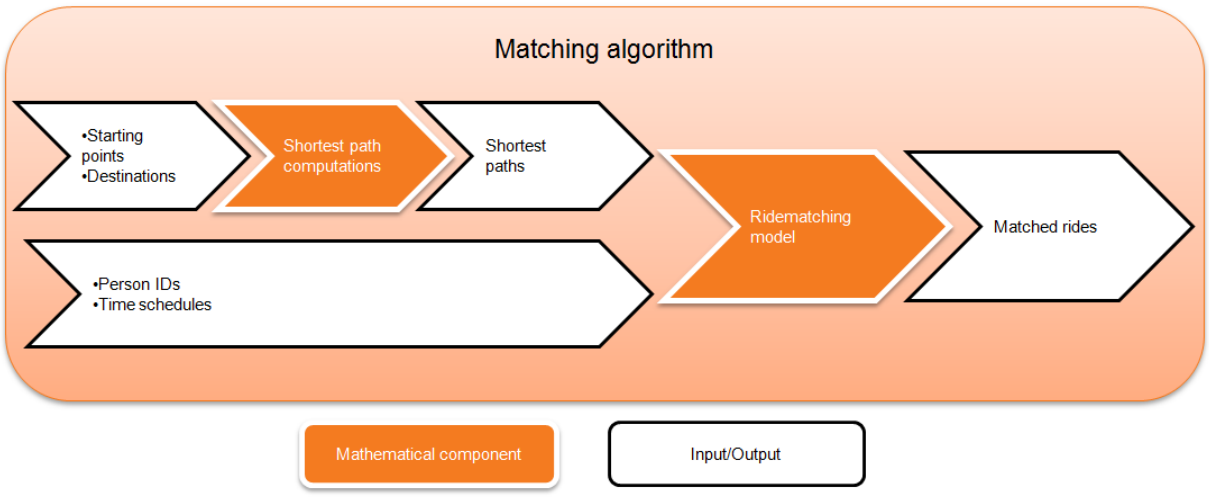


Figure 4-2: Underlying methods of the matching algorithm.

In Section 3.3, we discussed the ways to solve shortest path problems. We assume shortest path techniques are known and consider these computations as 'black box' computations in this chapter. This means that we insert a starting point and destination (as Figure 4-2 shows) in the 'black box'. Subsequently, the box returns a shortest path. The calculated shortest paths, paired with corresponding person ID, are input to the ridematching model.

Besides this, the time schedules of each user are given to the ridematching model. In Section 2.4, we suggested to derive inputs, such as time schedules, from agendas. For our ridematching model, we assume the starting times of appointments in an agenda are latest arrival times. Likewise, the finish times of appointments represent the earliest departure times. So inputs to our model are latest arrival times and earliest departure times.

The mathematical formulation of the shortest path problem in Section 3.3 serves as input to our ridematching model. For our model we consider the same graph $G = (V, E)$, with a set V of m vertices and a set E of n edges. A decision variable of the shortest path formulation is x_{pij}, which indicates whether the edge (i, j) is on the shortest path of person $p \epsilon P$ (with $x_{pij} = 1$ indicating edge (i, j) is part of the shortest path of person p and $x_{pij} = 0$ otherwise). This variable x_{pij} is used as a parameter in our ridematching model. The vertices of the graph represent only relevant places for a carpool model, which are starting points and destinations of the users' trips and possible meeting points for a carpool, such as Park and Rides. So graph $G = (V, E)$ is a transit node network with edges (i, j) connecting relevant vertices i and j. For the development of our ridematching model, we associate each edge (i, j) with travel time t_{ij}, which stands for the time it takes to drive from node i to node j. So a shortest path is the path from origin to destination that takes the least amount of time. Without loss of generality, we assume a linear relationship between time and distance. Subsequently, the shortest path is also the path that requires the least amount of travel distance.

So inputs to the ridematching model are:

- Shortest paths of users from their origin to destination, with parameter $x_{pij} = 1$ indicating edge (i, j) is part of the shortest path of person p and $x_{pij} = 0$ otherwise. Vertices i and j represent relevant carpool points, which can serve as starting points or finish points of a shared ride.
- Travel times t_{ij} between node i to j.
- Time schedules, representing the earliest departure times of the users from their origins and latest arrival times at their destinations.
- Persons IDs.

4.3 Model Development

In Section 4.1, we already described the ideas behind our ridematching model. The model should recognize overlapping routes and suggest carpools that neither for the passenger nor the driver lead to a detour. The carpools are enabled by the assumptions that all users have a car and are able to travel to a certain meeting point. For determining overlap, we only examine relevant nodes, where persons can start or end a carpool, as described in Section 4.2. So, overlap in routes exists when two persons travel both from the same node i to the same node j. However, in Section 4.1, we emphasized that carpooling an overlapping part of the route does not guarantee that all carpool participants reach their destinations. Therefore, the ridematching model should ensure that every person arrives at his destination, when undertaking a carpool. In this section, we develop this ridematching model. For the purpose of clarification, we build the ridematching model in three stages:

- *Basic Ridematching Model (Section 4.3.1)*
  The model filters overlap in routes and sets up carpools between persons with overlapping routes when the passenger is able to arrive at his destination with the set of offered rides. Passenger and driver can drive to a meeting point to start the carpool, and the passenger can hop between carpools.

- *Time & Capacity Extension (Section 4.3.2)*
  Besides the basic functionalities, the model filters overlap in time schedules and sets up carpools between persons with overlapping time schedules when the driver is able to pick up the passenger (capacity).
- *Return Restrictions (Section 4.3.3)*
  Besides the basic, time, and capacity functionalities, the model sets up carpools between persons, taking into account that each person is able to reach all his destinations in a certain time period and, when the car is parked at a meeting point, is able to return to his car on a later ride. Due to the return restrictions, the model has to cope with multiple rides per person in a certain time period.

The structure in Sections 4.3.1, 4.3.2, and 4.3.3 is as follows. First, we elaborate on the functionalities that the concerning stage offers to the ridematching model. After this, we introduce objectives and constraints, needed to adapt the model to these functionalities. Finally, we mathematically formulate these objectives and constraints.

4.3.1 Basic Ridematching Model

In this section, we develop the basic ridematching model, which filters overlap in routes and sets up carpools between persons with overlapping routes when the passenger is able to arrive at his destination with the set of offered rides. Passenger and driver can drive to a meeting point to start the carpool, and the passenger can hop between carpools. Section 4.3.1.1 describes the functionalities of the basic ridematching model, Section 4.3.1.2 discusses the objectives and constraints that these functionalities impose on the model, and Section 4.3.1.3 mathematically formulates these objectives and constraints.

4.3.1.1 Functionalities

The basic functionalities of the ridematching model consist of matching users with overlapping routes for a shared ride. Each user is assigned to one route, because the ridematching model is not able to cope with alternative routes. As described in Section 4.2, this one route is the user's shortest path. In Figure 4-3, the path of user i consists of nodes 1, 3, and 4, and the path of user j consists of nodes 2, 3, and 4. The routes of users i and j overlap on edge (3,4). We strive to minimize the number of self-driven edges by all users, so we prefer a solution where edge (3,4) is driven once over a solution where edge (3,4) is driven multiple times. Thus, the ridematching model matches users i and j for a carpool from node 3 to 4.

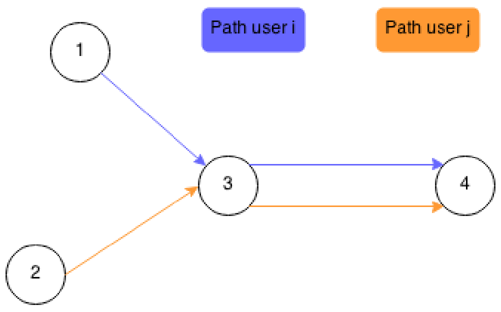


Figure 4-3: Start carpool at a meeting point.

In contrast to the scenario in Figure 4-3, a carpool does not have to start at a meeting point 'in the middle'. As Figure 4-4 shows, the ridematching model also suggests carpools to start at the origin of user j, when this origin is part of the shortest path of user i. Likewise, the ridematching model also suggests carpools when user i and j do not have the same destination. So, users i and j carpool from node 2 to 3 in Figure 4-4.

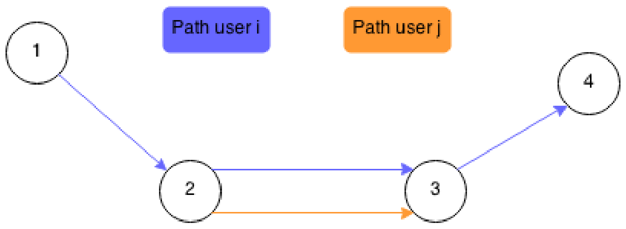


Figure 4-4: Start carpool at the starting point of user j.

The ridematching model is not restricted to one carpool per ride. In Figure 4-5, we show that user i participates in multiple carpools on his ride from node 1 to 7. First, user j rides with user i from node 3 to 4. Thereafter, user i carpools with user k from node 6 to 7. Because the ridematching model ensures every user reaches his destination, the model determines that user j has to leave his car at node 3, because user i needs his car to drive from node 4 to 6.

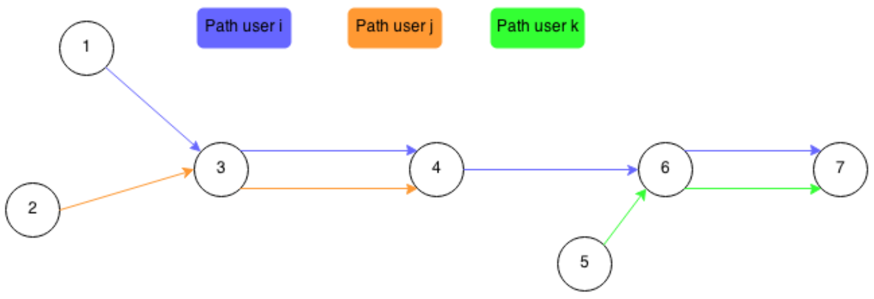


Figure 4-5: Multiple carpools per driver.

The ridematching model also enables users to hop between carpools, which is shown in Figure 4-6. The model matches users i and j to carpool from node 3 to 4, and users j and k to carpool from node 4 to 7. User j leaves his car at node 3, because user i needs his car to reach node 6. The ridematching model takes into account that user j cannot drive anymore, once he left his car at node 3. So, the model suggests that user j hops between carpools at node 4, to reach his destination node 7.

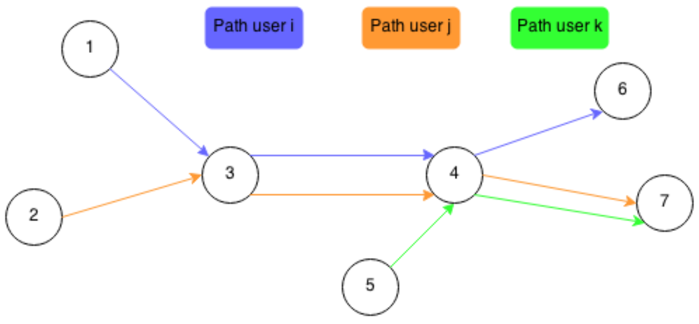


Figure 4-6: Multiple carpools per passenger.

In Figure 4-7, we show a comparable scenario to that of Figure 4-6. However, user j travels to node 8 and user k travels to node 9, in this case. Due to this adjustment, the ridematching model does not suggest any carpools, because user i needs his car to reach node 6, user j needs his car to reach node 8, and user k needs his car to reach node 9. As mentioned earlier, each user has to reach his destination and cannot use his car anymore, once he left the car to participate in a carpool. Although overlap in routes exists, the ridematching model does not recognize any carpool matches in the scenario of Figure 4-7.

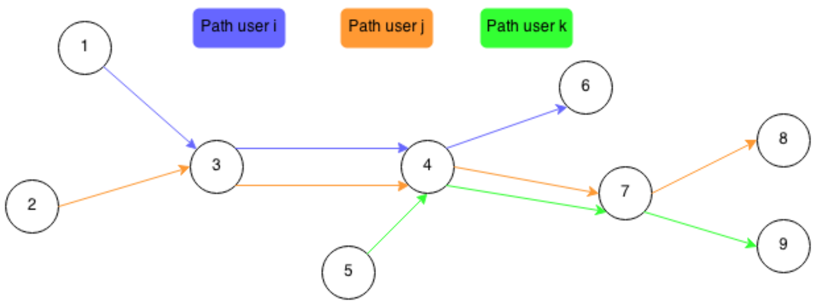


Figure 4-7: Overlap in routes, but no matches made by the model.

The possibility to hop between carpools does not necessarily mean that the passenger should actually switch between the carpools. In Figure 4-8, user j rides with user i on edge (3,4). Node 4 offers user j the opportunity to switch to a carpool with user k to carpool from node 4 to node 7. However, this is an unnecessary switch, because user i also drives edge (4,7) and users i and j are already in one car, because they carpooled on edge (3,4). The ridematching model prevents such unnecessary hops, and thus matches users i and j to carpool from node 3 to node 7.

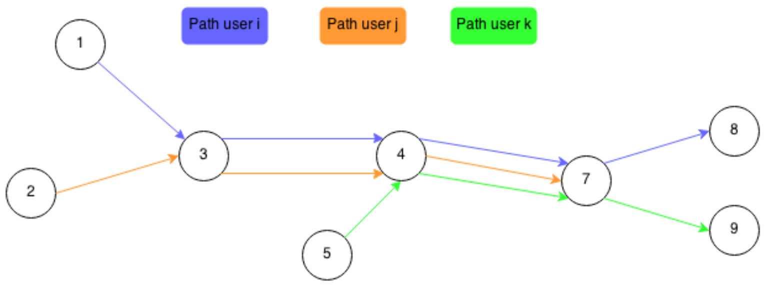


Figure 4-8: No unnecessary hops between carpools.

4.3.1.2 Objective & Constraints

In the discussion on the model's basic functionalities in Section 4.3.1.1, we already introduced the objectives of the ridematching model. We mentioned that we prefer a solution where an edge is driven once over a solution where this edge is driven multiple times, and that the model should prevent unnecessary hops. In this section, we elaborate on these objectives. Furthermore, we discuss the constraints that bound the model.

The main goal of the basic ridematching model is to maximize the total distance carpooled, which is the same as minimizing the total distance driven. This goal causes the model to prefer a solution where an edge is driven once over a solution where this edge is driven multiple times. As discussed in Section 4.2, we assume a linear relationship between distance and time. Hereby, the main objective becomes to minimize the time driven by all persons. In which 'time driven' means the time a person actually sits behind the wheel. The second objective of the basic ridematching model is to prevent unnecessary hops between carpools. This is realized by minimizing the number of carpools. This second goal of minimizing the number of carpools seems to conflict with the main goal of maximizing the total distance carpooled. So, it is important to tune these goals such that the model generates a solution with a maximum total distance carpooled without any unnecessary carpools.

So, in matching persons for shared rides, the model strives to minimize:

1. Total time spent as a driver for all persons
2. The number of carpools

In reaching the two objectives, the model is bounded by certain constraints. Based on the description of the model's basic functionalities in Section 4.3.1.1, we determine the following constraints:

- When carpooling, persons should have overlapping (shortest) paths. This is the basic idea of the ridematching model (see Section 4.1).
- When a person leaves his car for a shared ride, he cannot drive anymore. The person parks his car at a parking lot and continues his trip in someone else's car.
- Each person should arrive at his destination. Carpooling is not possible, when it does not enable persons to reach their destinations.
- If person a rides with person b on a certain trajectory, then person b cannot ride with someone else on this trajectory. Meaning that when person b is the driver of a carpool between persons a and b, then person b cannot sit in the passenger's seat of another car on the same trajectory.

In Section 4.3.1.3, we provide the mathematical formulation of the objectives and constraints of the basic ridematching model, as discussed in this section.

4.3.1.3 Mathematical Formulation

In this section, we mathematically formulate the basic ridematching model, which is capable of performing the functionalities described in Section 4.3.1.1. In our formulation we use two decision variables:

- Z_{pij}, a binary variable indicating whether person p drives from i to j. So $Z_{pij} = 1$ when person p sits in the driver's seat from i to j and $Z_{pij} = 0$ otherwise.
- Y_{pqij}, binary variable indicating whether person p rides with person q from i to j. So $Y_{pqij} = 1$ when person p sits in the passenger's seat and person q sits in the driver's seat from i to j, and $Y_{pqij} = 0$ otherwise.

Furthermore, we introduce a dependent variable, which depends on the outcomes of Z_{pij} and Y_{pqij}:

- S_{pqi}, a binary variable indicating the starting point i of the carpool, in case person p carpools with person q. $S_{pqi} = 1$ when person p meets person q for a carpool at node i, and $S_{pqi} = 0$ otherwise.

To recapitulate Section 4.2, we make use of two parameters for the basic ridematching model:

- x_{pij}, a parameter with $x_{pij} = 1$ indicating edge (i, j) is part of the shortest path of person p and $x_{pij} = 0$ otherwise. Vertices i and j represent relevant carpool points, which can serve as starting points or finish points of a shared ride.
- t_{ij}, a parameter representing the travel time t_{ij} between node i to j.

The objective of the basic ridematching model is to minimize the time driven by all persons and the number of carpools needed to achieve this, see Section 4.3.1.2. We use the starting point of a carpool as indicator for a carpool, so the sum of all carpool starting points is equal to the number of carpools. Because we enumerate two objectives with different units of measurement and not every objective is equally important, we introduce weights for each objective. By assigning much more weight to the objective that minimizes the time driven than to the objective that minimizes the number of carpools, we can modify the model such that sufficiently long carpools are not outweighed by the number of carpools. We strive for a solution with a minimum amount of time driven. When multiple ridematching schemes exist that generate a minimum amount of time driven, the model should prefer the scheme with the least amount of carpools. So, the objective function becomes:

$$Minimize\ w_1 * \sum_{p \in P} \sum_{i,j \in V} t_{ij} * Z_{pij} + w_2 * \sum_{p,q \in P} \sum_{i \in V} S_{pqi}$$

Subject to

(1)	$Y_{ppij} = 0$	$\forall p \in P, \forall i,j \in V$
(2)	$\sum_{q \in P} Y_{pqij} + Z_{pij} = x_{pij}$	$\forall p \in P, \forall i,j \in V$
(3)	$\sum_{q \in P} \sum_{i \in V} Y_{pqij} + \sum_{i \in V} Z_{pji} \leq 1$	$\forall p \in P, \forall j \in V$
(4)	$Y_{pqij} \leq Z_{qij}$	$\forall p,q \in P, \forall i,j \in V$
(5)	$S_{pqj} \geq \sum_{i \in V} Y_{pqji} - \sum_{i \in V} Y_{pqij}$	$\forall p,q \in P, \forall j \in V$
(23)	$Y_{pqij}, Z_{pij}, S_{pqi} \in \{0,1\}$	$\forall p,q \in P, \forall i,j \in V$

We use a remarkable numbering, see constraint (23), to anticipate the addition of more constraints to the model in Sections 4.3.2.3 and 4.3.3.3. We add the new constraints between constraint (5) and constraint (23).

The first constraint (1) precludes the possibility of persons carpooling with themselves. Constraint (2) is used to make sure that a person p cannot carpool and drive on the same edge (i,j), and that a person can only drive or carpool on edges of his shortest path. Also, because of constraint (2) a person p can only ride with one person q on edge (i,j), due to the summation over q for Y_{pqij} and the fact that x_{pij} is at most 1, see (23). When a person p rides with person q on edge (i,j), person p leaves his car at node i and therefore cannot drive anymore for the rest of his trip, which is coped by constraint (3). Together, constraints (2) and (3) guarantee arrival at the destination, because all edges of the shortest path have either to be driven or to be carpooled (2), and in case of carpooling, person p has to continue carpooling until he reaches his destination (3). Constraint (4) secures that person q drives, when person p rides with person q. Constraint (5) determines at which node the carpool starts. If person p rides with person q, S_{pqj} is set to 1 when the carpool starts at node j, and set to 0 otherwise. Constraint (23) restricts variables to be either 1 or 0.

In the beginning of this section, we mention that we make use of the parameter x_{pij} with $x_{pij} = 1$ indicating edge (i,j) is part of the shortest path of person p and $x_{pij} = 0$ otherwise. However, when edge (i,j) is not part of the shortest path of person p, we do not give any value of x_{pij} to the model. In this situation, the edge (i,j) in relation to person p, does not exist in the model. Consequently, if $x_{pij} = 0$, then other variables, such as Z_{pji} and Y_{pqij} (for any q), do not exist in the model. Due to this 'trick', we limit the model to only consider relevant edges, which saves running time. For example, when the shortest path of person p consists of 4 edges, while the entire network consists of 1000 edges, the model has to consider constraint (2) only 4 times, instead of 1000 times. This means that the expression $\forall p \in P, \forall j \in V$, after constraint (3), can be read as: for all nodes j on the shortest path of person p.

The model we presented in this section is only a basic model for carpooling amongst persons that drive the same route. The model filters overlap in routes and sets up carpools between persons with overlapping routes when the passenger is able to arrive at his destination with the set of offered rides. However, this model does not yet include constraints concerning time and capacity, nor does it cope with return restrictions. In Section 4.3.2, we design the time and capacity constraints, and thereafter, in Section 4.3.3, the return restrictions.

4.3.2 Time & Capacity Extension

In this section, we extend the basic ridematching model with time and capacity constraints. With this extension, the model filters overlap in time schedules and sets up carpools between persons with overlapping time schedules when the driver is able to pick up the passenger (capacity). Section 4.3.2.1 describes the functionalities of the extended model, Section 4.3.2.2 discusses the objectives and constraints that these functionalities impose on the model, and Section 4.3.2.3 mathematically formulates these objectives and constraints.

4.3.2.1 Functionalities

In Section 4.3.1.1, we present the basic functionalities of the ridematching model. These functionalities enable the model to match users with overlapping paths, while ensuring each user reaches his destination. In order to match users for a carpool, however, their time windows have to correspond as well, as stated in Section 1.4. In this section, we discuss the ridematching model that takes into account overlapping time windows and car capacity. This model is an extension of the basic ridematching model, so functionalities discussed in Section 4.3.1.1 are still applicable.

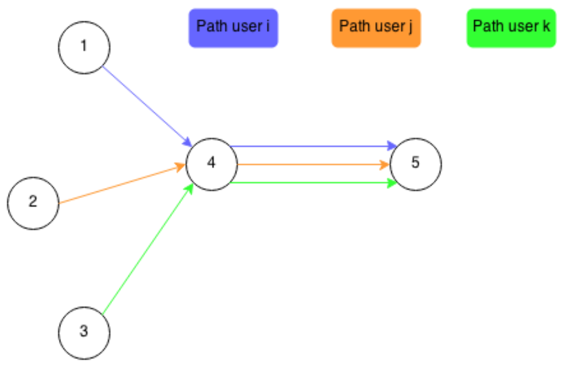


Figure 4-9: Multiple passengers in one carpool simultaneously.

The outcome of the ridematching model depends on the capacity (seats available) set by the users. In the capacity, the driver’s seat is not taken into account, so a capacity of 2 means that the driver can take at most 2 passengers (so at most 3 persons in one car). Figure 4-9 shows a situation where the routes of three users come together at node 4. When all users have a capacity of 0, the ridematching model will not suggest any carpools. In the situation where at least one of the users has a capacity of 2 or more, the ridematching model suggests that two users leave their car at node 4, because all three users can continue their trip in one car (with a capacity of 2 or more). For example, this situation occurs when the capacity of user i is 2 and the capacity of users j and k is 1. Because the cars of users j and k do not have enough seats available to transport all users simultaneously from node 4 to 5, the ridematching model will recommend that these users leave their car at node 4 and continue their trip in the car of user i. With this solution, edge (4,5) is only driven once. As discussed in Section 4.3.1.1, we prefer a solution where edge (4,5) is driven once over a solution where edge (4,5) is driven multiple times. So in any case, the ridematching model gives a suggestion with a minimum amount of edges driven within capacity boundaries of all users.

We illustrate paths of users i and j with corresponding travel times in Figure 4-10. In Section 4.2, we assumed that inputs to our model are latest arrival times and earliest departure times. These times are hard constraints, which means that a person cannot arrive later at his destination than the latest arrival time and cannot leave earlier from his origin than the earliest departure time. By describing three scenarios (see Table 4-1), we elaborate on the time constraints in the model.

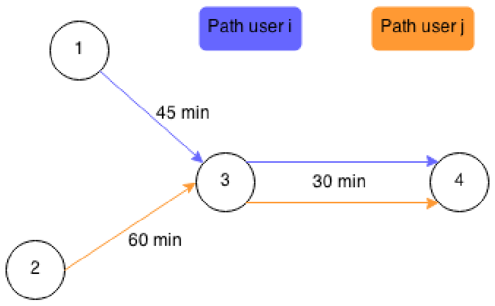


Figure 4-10: Network including travel times.

	User	Latest arrival time	Earliest departure time
Scenario 1	i	09:00	07:00
	j	09:00	07:00
Scenario 2	i	09:00	07:00
	j	09:15	07:00
Scenario 3	i	08:15	07:00
	j	09:00	07:00

Table 4-1: Three different scenarios concerning latest arrival time and earliest departure time.

- *Scenario 1*
  The latest arrival time of both users is 09:00 at node 4. The model plans arrivals as close as possible to their latest arrival time. So, the users arrive at 09:00 at node 4. This means user i has to depart from node 1 at 07:45, and user j has to depart from node 2 at 07:30 (see Figure 4-10). These departure times are later than the earliest departure times, so the ridematching model will suggest that user i meets user j at node 3 at 08:30 to undertake a carpool on edge (3,4).
- *Scenario 2*
  Since the latest arrival time of user i is 15 minutes earlier than the latest arrival time of user j, a carpool is only possible when user j is able to arrive at 09:00 at node 4. In order to carpool on edge (3,4) and arrive at 09:00 at node 4, user i has to depart from node 1 at 07:45, and user j has to depart from node 2 at 07:30. These departure times are later than the earliest departure times, so the ridematching model will suggest that user i meets user j at node 3 at 08:30 to undertake a carpool on edge (3,4).
- *Scenario 3*
  Since the latest arrival time of user j is 45 minutes later than the latest arrival time of user i, a carpool is only possible when user j is able to arrive at 08:15 at node 4. In order to carpool on edge (3,4) and arrive at 08:15 at node 4, user i has to depart from node 1 at 07:00, and user j has to depart from node 2 at

06:45. This departure time of user j is earlier than his earliest departure time, so the ridematching model does not generate a carpool match.

In the three scenarios, the total trip duration remains unchanged for both users. In all scenarios the total trip duration for user i is 75 minutes, and for user j 90 minutes. The model only varies their departure and arrival times within boundaries (earliest departure time and latest arrival time). In scenarios 1 and 2, users i and j carpool from node 3 to 4 without incurring any delay. However, in some situations the model does allocate waiting time at an intermediate node to the users. In Figure 4-11, we assume users i and j carpool on edge (3,4) and users k and l carpool on edge (8,4). Furthermore, users j and k share a ride on edge (4,5). But when the arrival time of users i and j at node 4 is, for example, 11:30, and the arrival time of users k and l at node 4 is 12:00, then user j has to wait 30 minutes for user k at node 4. Assuming that these carpools are possible within latest arrival times and earliest departure times of all users, the model suggests that user j has to wait 30 minutes at node 4. In addition to this, we make three comments:

- With waiting time we mean the difference between the actual trip duration and the fastest trip duration. So, we only consider waiting times at intermediate nodes between starting point and destination as waiting time. A deviation from earliest departure time at the starting point or from the latest arrival time at the destination is not considered as waiting time.
- The model tries to reduce waiting time to a minimum. The ridematching model tries to adjust departure times of all users (within boundaries) such that the waiting time at an intermediate node is reduced to a minimum.
- Matches are based on a maximum waiting time. The model takes into account a maximum total delay a user may incur en route. So, if the total maximum delay a user may incur is less than 30 minutes, the ridematching model does not match users j and k for a carpool, because user j has to wait 30 minutes at node 4 in Figure 4-11.

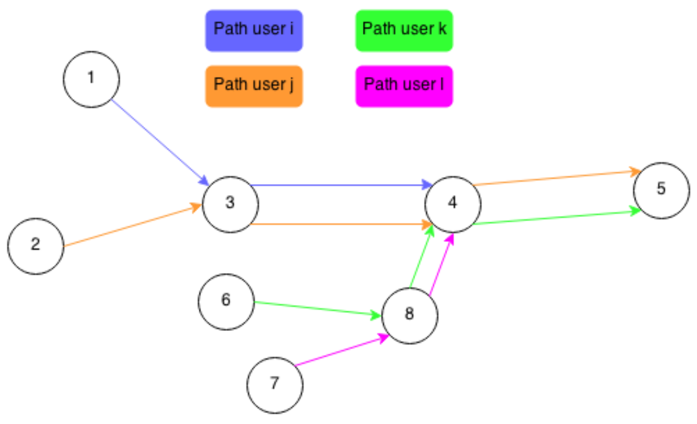


Figure 4-11: Waiting time at node 4.

4.3.2.2 Objective & Constraints

Section 4.3.2.1 discussed the ridematching model that takes into account overlap in time windows and car capacity, which is an extension of the basic ridematching model. We state that the extended model plans arrivals as close as possible to their latest arrival time, and tries to reduce waiting time to a minimum. This means that the extended model requires extra objectives, compared to the basic ridematching model. In this section, we elaborate on the required extra objectives and constraints.

As discussed in Section 4.3.1.2, the basic ridematching model consists of two goals. The first goal is to minimize the total time spent as a driver by all persons, and the second goal is to minimize the number of carpools needed to achieve the first goal. In order to cope with overlap in time windows, we add a third and fourth goal to the basic ridematching model. The third goal of the model is to plan arrivals as close as possible to their latest arrival time, so minimizing the difference between the latest arrival times and the actual arrival times. We assume that a person, when driving alone to an appointment, always aims at arriving at the latest arrival time. Our ridematching model should lead to arrival times that minimally deviate from latest arrival times to keep time schedules largely the same as in the situation where people do not carpool. As mentioned in Section 4.3.2.1, a person cannot arrive later than his latest arrival time. Constraints should ensure that the person does not arrive too late. The fourth and final goal is to reduce waiting times to a minimum. In Section 4.3.2.1, we stated that the model reduces waiting times by adjusting departure (and thus arrival) times, within boundaries. So, just as with the first and second goal, the fourth goal, minimizing waiting times, conflicts with the third goal of planning arrivals as close as possible to their latest arrival time. Similarly, the third and fourth goal have to be tuned such that the model generates a solution with a minimum amount of total waiting time without any unnecessary deviation from latest arrival times. In conclusion, the goals of minimizing the number of carpools and minimizing deviation from latest arrival times are subordinate to the main goals of maximizing the total distance carpooled and minimizing the total waiting time.

So, the extended ridematching model, with the ability to match on overlap in time windows, is divided in four objectives. In matching persons for shared rides, the model strives to minimize:

1. Total time spent as a driver by all persons
2. The number of carpools
3. Deviation of arrival time at destination from latest arrival time
4. Total waiting time of all persons

In which the second and third objectives are inferior to the first and fourth objectives. So, not every objective is equally important. Therefore, weights have to be assigned to each objective.

In Section 4.3.1.2, we already discussed the constraints of the basic ridematching model. The functionalities discussed in Section 4.3.2.1 impose further constraints on the model:

- On each trajectory of his shortest path, the driver of a carpool cannot take more passengers than the set capacity.
- For each ride, the total waiting time per person cannot exceed a given threshold value (maximum waiting time).
- Persons cannot arrive later than latest arrival times and cannot depart earlier than earliest departure times.
- Persons need to have equal departure and arrival times in case of carpooling. As shown in Section 4.3.2.1, persons with unequal latest arrival times can be matched by the model. The model adjusts their departure times at the starting point of the carpool and arrival times at the destination of the carpool (in scenario 2 from Section 4.3.2.1, the time schedule of user j is brought 15 minutes forward).

In Section 4.3.2.3, we provide the mathematical formulation of the objectives and constraints of the ridematching model that takes into account overlap in time windows and car capacity, as discussed in this section.

4.3.2.3 Mathematical Formulation

In this section, we expand the basic ridematching model of Section 4.3.1.3 with time and capacity constraints. Persons can only undertake a carpool when they have comparable time schedules, and the person driving cannot take an unlimited amount of people due to the size of the car. We discussed the functionalities of this extended ridematching model in Section 4.3.2.1. Section 4.3.2.2 described the consequences that these functionalities have on the model, in the form of objectives and constraints. In this section, we translate those objectives and constraints into mathematical formulations.

First, we discuss the formulation of the capacity constraint. In Section 4.3.2.1, we define the capacity as the seats available to passengers in a person's car (driver's seat is not taken into account). So, we assume each person p indicates how much passengers he can, or is willing to take simultaneously, indicated by Cap_p. Below we show the capacity constraint that can be added to the model. The constraint is rather simple and ensures that for every edge (i,j), the number of riders in the car of person q does not exceed the maximum capacity given by person q.

(6) $$\sum_{p \in P} Y_{pqij} \leq Cap_q \qquad \forall q \in P, \forall i,j \in V$$

Next, we formulate the time constraints. For this, we define des_p as the destination of person p, and or_p as the origin of person p. Other inputs are the latest arrival time l_p of person p at his destination, the earliest departure time e_p of person p from his origin, and the maximum waiting time m_p during the ride for person p. This maximum waiting time m_p is determined as the sum of the waiting times on all intermediate nodes of person p. Furthermore we introduce a new decision variable D_{pi}, which is departure time of person p at node i. We also suggest a variable that depends on D_{pi}, namely A_{pi}, which stands for the arrival time of person p at node i.

In Section 4.3.2.2, we defined the model's objective concerning time, which is to minimize total waiting time without any unnecessary deviation from latest arrival times. Outcomes of the model are constrained by a maximum waiting time per person, latest arrival time at destination, earliest departure time from origin and equalization of departure times and arrival times of persons that carpool together. We define the arrival time of person p at node j as the departure time of person p at node i plus the travel time from node i to j:

$$A_{pj} = D_{pi} + t_{ij}$$

Under the condition that the departure time at node i for person p is always larger than, or equal to his arrival time at node i, the waiting time of person p at node i is equal to the difference between the departure time D_{pi} of person p at node i and the arrival time A_{pi} of person p at node i. Subsequently, the total waiting time incurred by person p during the ride, is defined as the difference between the actual trip duration and the fastest trip duration (see Section 4.3.2.1). The actual trip duration is equal to the difference between the arrival time A_{p,des_p} of person p at his destination and the departure time D_{p,or_p} at his starting point, and the fastest trip duration is equal to the sum of the travel time of all edges on his shortest path, therefore total waiting time of person p is:

$$\left(A_{p,des_p} - D_{p,or_p}\right) - \sum_{i,j \in V} t_{ij} * x_{pij}$$

The deviation from the latest arrival time for person p is defined as the difference between latest arrival time at his destination l_p and the actual arrival time at his destination A_{p,des_p}. The model's time-related objective is to minimize total waiting time, while avoiding unnecessary deviation from latest arrival times, see Section 4.3.2.2. Therefore we add two new objectives to the basic ridematching model. The extended ridematching model also strives to minimize total deviation from latest arrival time and total waiting time for all persons, such that the new objective function becomes:

$$Minimize \left(\begin{array}{c} w_1 * \sum_{p \in P} \sum_{i,j \in V} t_{ij} * Z_{pij} + w_2 * \sum_{p,q \in P} \sum_{i \in V} S_{pqi} + w_3 * \sum_{p \in P} \left(l_p - A_{p,des_p} \right) + \\ w_4 * \sum_{p \in P} \left(\left(A_{p,des_p} - D_{p,or_p} \right) - \sum_{i,j \in V} t_{ij} * x_{pij} \right) \end{array} \right)$$

New constraints added to the model are:

(7)	$A_{pj} = D_{pi} + t_{ij}$	$\forall p \in P, \forall i,j \in V$
(8)	$D_{pi} \geq A_{pi}$	$\forall p \in P, \forall i \in V$
(9)	$A_{p,des_p} \leq l_p$	$\forall p \in P$
(10)	$D_{p,or_p} \geq e_p$	$\forall p \in P$
(11)	$D_{pi} - D_{qi} \leq M * \left(1 - Y_{pqij}\right)$	$\forall p,q \in P, \forall i,j \in V$
(12)	$D_{pi} - D_{qi} \geq -M * \left(1 - Y_{pqij}\right)$	$\forall p,q \in P, \forall i,j \in V$
(13)	$\left(A_{p,des_p} - D_{p,or_p}\right) - \sum_{i,j \in V} t_{ij} * x_{pij} \leq m_p$	$\forall p \in P$
(23) *addition*	$A_{pi}, D_{pi} \geq 0$	$\forall p \in P, \forall i \in V$

Constraint (7) defines the arrival time at node j of person p as the departure time of person p at node i plus the travel time from node i to j. Constraint (8) determines that the departure time at node i for person p is always greater than, or equal to his arrival time at node i. Constraint (9) restricts the arrival time of person p at his destination to be smaller than the latest arrival time l_p, while constraint (10) restricts the departure time of person p at his origin to be bigger than the earliest departure time e_p. When person p rides with person q from node i to node j, their departure times at node i have to be equal, which is ensured by constraints (11) and (12). The model minimizes the total waiting time for all persons, but constraint (13) sets a maximum waiting time m_p per person. Constraint (23) determines that the departure time or arrival time of person p can take any positive value.

Similarly to the 'trick' we applied to the model of Section 4.3.1.3, we limit the model to only consider relevant edges. For constraint (7), this means that we only calculate arrival times for those nodes j, by summing departure time at node i and the travel time from i to j, when edge (i,j) is on the shortest path of person p. So, $\forall p \in P, \forall\, i,j \in V$ means: for all p,i,j for which holds $x_{pij} = 1$. All possible edges that are not on the shortest path of person p are not considered.

4.3.3 Return Restrictions

In this section, we add return restriction to the model of Section 4.3.2. With this extension, the model sets up carpools between persons, when each person is able to reach all his destinations in a certain time period and, when the car is parked at a meeting point, is able to return to his car on a later ride. Due to the return restrictions, the model has to cope with multiple rides per person in a certain time period. Section 4.3.3.1 describes the functionalities of the model with return restrictions, Section 4.3.3.2 discusses the objectives and constraints that these functionalities impose on the model, and Section 4.3.3.3 mathematically formulates these objectives and constraints.

4.3.3.1 Functionalities

Section 4.3.2.1 presented the functionalities of the ridematching model that takes into account overlapping time windows and car capacity, which is an extended version of the basic ridematching model (see Section 4.3.1.1). In this section, we further extend the ridematching model by adding return restrictions, which means that the model only sets up carpools between persons, when each person is able to reach all his destinations in a certain time period and, when the car is parked at a meeting point, is able to return to his car on a later ride. Due to the return restrictions, the model has to cope with multiple rides per person in a certain time period.

In the left part of Figure 4-12, users i and j share a ride from node 3 to 4, so one user leaves his car at node 3. However, the extended ridematching model, with return restrictions, takes into account all rides of the users in a certain period. For example, not only the ride from home to work, but also the ride from work to home. Return restrictions enable the model to recognize whether a user returns to a certain node on a later ride that day. We illustrate this in Figure 4-12. User j passes node 3 on ride r, and returns to node 3 on ride w. The model determines that user i rides with user j on ride r from node 3 to 4, and with user h on ride w from node 11 to 3. So, compared to the first ride r, the return ride w does not have to be shared with the same person (h instead of i), and does not have to contain the same edges (user j visits (11,3) and (3,12) on the return ride). Logically, when neither user j nor user i passes node 3 on a later ride, or can carpool back to node 3, the ridematching model does not generate a match between users i and j on ride r.

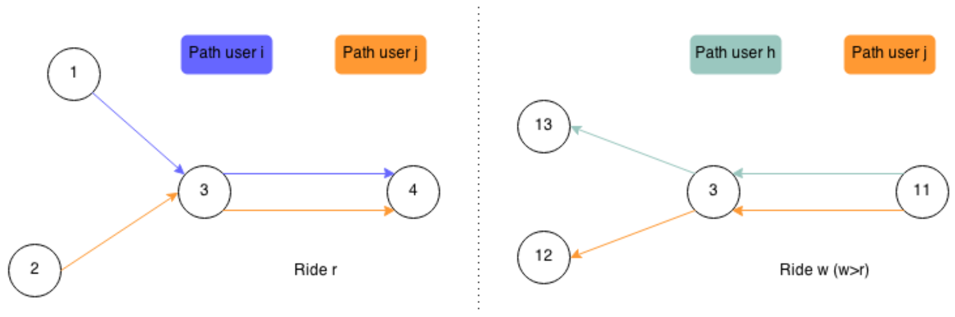


Figure 4-12: User j leaves his car at node 3 to join a shared ride.

4.3.3.2 Objective & Constraints

In Section 4.3.3.1, we discussed the ridematching model with return restrictions, which is an extension of the ridematching model of Section 4.3.2. This extension does not lead to any extra objectives of the model, it only imposes two extra constraints on the model, which we discuss in this section.

The ridematching mode, that we introduced in Section 4.3.3.1, is able to take into account multiple rides per person and to recognize whether a user returns to a certain node on a later ride that day. This functionality adds two constraints to the model, namely:

- A person can only leave his car at a certain node to join a carpool, when this person returns to this node on a later ride.
- When a person leaves his car at a meeting point, he has to travel by carpooling, until he returns to this meeting point.

In Section 4.3.2.3, we provide the mathematical formulation of the constraints of the ridematching model with return restrictions, as discussed in this section.

4.3.3.3 Mathematical Formulation

In Section 4.3.2.3, we defined a complete mathematical formulation of the method we introduced in Section 4.1. However, this ridematching model is only suitable for matching one-way rides, which is not a realistic situation. When the model determines that a person has to leave his car at some place, this person should be able to return to his car in a later ride. In this section, we add the functionalities described in Section 4.3.3.1.

In Section 4.3.3.1, we stated that "*return functionalities enable the model to recognize whether a user returns to a certain node on a later ride that day*". So the model should take into account multiple rides per person per day. For this, we suggest a new set, namely the set of rides R. For example, an employee starts and ends his day at home and visits two clients during the day. Then his first ride of the day is from home to client 1, the second ride is from client 1 to client 2, and the last ride of the day is from client 2 to his home. So, the set of rides R for this employee consists of three rides {1,2,3}. Each variable and parameter that depends on the set of persons P, now also depends on the set of rides R, such that the value of the variable or parameter not only depends on the person, but also on the ride number of that person. For example, Z_{pij} becomes Z_{prij}, indicating whether person p drives from node i to j on ride r, and D_{pj} turns to D_{prj}, indicating the departure time of person p at node j on ride r. Furthermore, we define a new (dependent) variable that indicates the 'certain node' where the user has to return to:

- C_{pri}, a binary variable indicating the node i where the car of person p is parked at the end of ride r. $C_{pri} = 1$ when the car of person p is parked on node i at the end of ride r, and $C_{pri} = 0$ otherwise.

With this variable we can steer the model such that a person comes back to the place where he parked his car. But first we explain how we determine the value of C_{pri}, that is, at the end of ride r, at which node i the car of person p is parked. Several possibilities exist, the car can be at:

- the destination of person p on ride r, in case person p did not carpool on ride r. Another option is that person p carpooled to his car, which he left on an earlier ride, and drove to the destination. Besides this, it is possible person p carpooled to the destination of ride r, while his car was still parked there due to an earlier carpool.
- a meeting point along the route, in case person p rides with someone else from this meeting point on ride r to the destination.
- the same place as the ride before, in case person p carpools the entire ride r. Person p does not drive on ride r, so his car can be at the origin of ride r (= destination $r - 1$) or earlier rides, or at a carpool meeting point of an earlier ride.
- home, in case person p is picked up at his first origin (origin of $r = 1$).

Five constraints have to ensure that C_{pri} indicates the right place when one of the above cases occurs.

(14) $$C_{pri} \leq 1 - \sum_{j \in V} Z_{prij} \qquad \forall p \in P, \forall r \in R, \forall i \in V$$

(15) $$C_{pri} \geq \sum_{j \in V} Z_{prji} - \sum_{j \in V} Z_{prij} \qquad \forall p \in P, \forall r \in R, \forall i \in V$$

(16) $$C_{pri} \geq C_{p,r-1,i} - \sum_{j \in V} Z_{prij} \qquad \forall p \in P, \forall r \in R, \forall i \in V$$

(17) $$C_{pri} \leq \sum_{j \in V} Z_{prji} + C_{p,r-1,i} \qquad \forall p \in P, \forall r \in R, \forall i \in V$$

(23) *addition* $$C_{pri} \in \{0,1\} \qquad \forall p \in P, \forall r \in R, \forall i \in V$$

Figure 4-13 provides a simple view of node i in relation to the mathematical formulations of person p driving <u>to</u> node i on ride r (left side) and person p driving <u>from</u> node i on ride r (right side). When person p drives from node i on ride r, then the car of person p cannot be parked on node i at the end of ride r, because he drove away from that node, so C_{pri} should be 0. This is captured by constraint (14). Constraint (15) ensures that when person p drives to node i on ride r, but does not drive from node i on ride r, the car of p is parked on node i at the end of ride r. So, when a person continues his trip by riding with someone else from node i, his car is parked at node i at the end of ride r. Also, when node i is the destination of person p on ride r and p drives to his destination (and naturally does not drive from this destination on ride r), the car is parked at node i at the end of ride r. It can occur that person p parks his car at node i on ride r and subsequently carpools the entire ride $r+1$, then at the end of ride $r+1$, the car is still parked at node i. Constraint (16) secures this situation. Once the car is picked up from this node (for example during ride $r+3$), constraint (14) sets $C_{p,r+3,i}$ back to 0. Constraint (17) determines that the car cannot be parked at node i at the end of ride r, if person p did not drive to node i on ride r, and the car is not parked there on an earlier ride. In case person p gets picked up at his first origin (origin of $r = 1$), the car stays at home. Due to constraint (17), the car is not parked at any node ($C_{pri} = 0$ for all nodes), in this case. Constraint (23) determines that C_{pri} is a binary variable, and thus can only be 1 or 0.

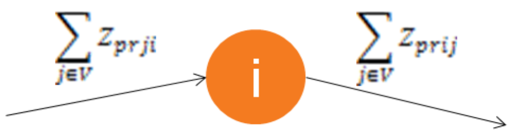


Figure 4-13: Node i where person p possibly parks his car

Constraints (15) to (17) determine where a person parked his car, at the end of a ride. With this information we can modify the model such that a person always picks up a parked car, otherwise the car cannot be parked. To secure these return restrictions, we modify constraint (3) and add new constraints to the model.

(3) *modified* $$\sum_{q \in P} \sum_{i \in V} Y_{pqrij} + \sum_{i \in V} Z_{prji} \leq 1 + C_{p,r-1,j} \qquad \forall p \in P, \forall r \in R, \forall j \in V$$

(18) $$\sum_{w > r} \sum_{j \in V} Z_{pwij} \geq C_{pri} \qquad \forall p \in P, \forall r \in R, \forall i \in V$$

(19) $$\sum_{q \in P} \sum_{i \in V} Y_{p,q,r,i,des_{pr}} + \sum_{i \in V} Z_{p,r+1,des_{pr},i} \leq 1 + C_{p,r,des_{pr}} \qquad \forall p \in P, \forall r \in R$$

Constraint (3) from the basic ridematching model is modified to cope with return restrictions. As described in Section 4.3.1.1, the constraint is designed to prevent persons from driving once they left their cars to join a carpool. However, with return restrictions one exception arises, namely when persons carpool to node j where they left their cars. Only then it is possible to carpool to node j and drive from node j. This situation is served by the modified version of constraint (3). In constraint (18), we state that if, at the end of ride r, the car of person p is parked at node i, then on a later ride ($w > r$), person p has to drive from node i to any node j, because he has to pick up his car. When person p does not drive from node i to any node j on a later ride ($\sum_{w>r} \sum_{j \in V} Z_{pwij} = 0$), then person p cannot park his car at node i on ride r ($C_{pri} = 0$). Constraint (19) is comparable with the modified constraint (3), but now we determine that person p cannot carpool to the destination of ride r and subsequently leave this node (in $r + 1$) with a car, unless he has parked his car at this location ($C_{pr,des_{pr}} = 1$).

As mentioned in the beginning of this section, to make return restrictions possible, the model should take into account multiple rides per person per day. Besides the introduced constraints in this section, these return restrictions have further consequences for the ridematching model. By introducing a set of rides for each person, the model's complexity rises. In the basic ridematching model (with time and capacity constraints), person p is simply matched with person q. Due to the dependency on the set of rides, each ride of person p can be matched with each ride of person q, see Figure 4-14.

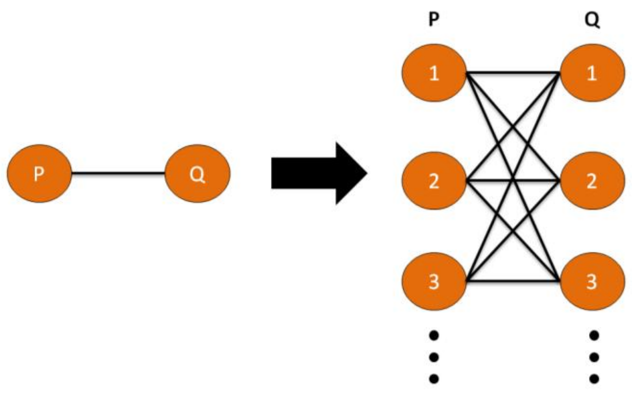


Figure 4-14: Increased complexity of ridematching model by introducing a set of rides for each person.

Constraint (4) has to be modified to handle this added complexity:

(4) *modified* $$Y_{pqrij} \leq \sum_{w \in R} Z_{qwij} \qquad \forall p, q \in P, \forall r \in R, \forall i, j \in V$$

The modified constraint (4) ensures that person p can only ride with person q from i to j on ride r (of person p), when person q does drive from i to j on any of his rides. The constraint enables that persons p and q can undertake a carpool. For example, the first ride of person p can match with the second ride of person q. To define which persons carpool on which edge, including their ride numbers, we introduce a new dependent variable:

- L_{pqrwij}, binary variable indicating whether person p rides with person q from i to j on ride r of p and ride w of q. So $L_{pqrwij} = 1$ when, on ride r of person p, person p sits in the passenger's seat and, on ride w of q, person q sits in the driver's seat from i to j, and $L_{pqrwij} = 0$ otherwise.

To define L_{pqrwij}, we add new constraints to the model:

(20) $$L_{pqrwij} \leq Y_{pqrij} \qquad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V$$

(21) $$L_{pqrwij} \leq Z_{qwij} \qquad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V$$

(22) $$\sum_{w \in R} L_{pqrwij} = Y_{pqrij} \qquad \forall p,q \in P, \forall r \in R, \forall i,j \in V$$

(23) *addition* $$L_{pqrwij} \in \{0,1\} \qquad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V$$

Constraints (20) to (22) define the (dependent) variable L_{pqrwij}, that provides insights into which persons join a carpool, on what edge they carpool, and the corresponding ride numbers of the persons during the carpool. Constraints (20) and (21) determine that persons p and q can only carpool on edge (i,j), on ride r of person p and ride w of person q, when person p carpools with person q on edge (i,j) during ride r of person p, and when person q drives on edge (i,j) during ride w of q. Furthermore, constraint (22) ensures that a carpool between persons p and q on edge (i,j) during ride r of person p, matches with only one ride w of person q. Constraint (23) determines that L_{pqrwij} is a binary variable, and thus can only be 1 or 0.

The variable L_{pqrwij} enables us to match persons with overlapping routes and time schedules, even though their ride numbers do not correspond. To determine whether time schedules show overlap, we defined the time constraints in Section 4.3.2.3. We have to adjust constraints (10) and (11) to be able to compare the time schedule of each ride of person p with the time schedule of each ride of person q. Furthermore, we modify constraint (6) to meet capacity restrictions of person q.

(6) *modified* $$\sum_{p \in P} \sum_{r \in R} L_{pqrwij} \leq Cap_q \qquad \forall q \in P, \forall w \in R, \forall i,j \in V$$

(10) *modified* $$D_{pri} - D_{qwi} \leq M * \left(1 - L_{pqrwij}\right) \qquad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V$$

(11) *modified* $$D_{pri} - D_{qiw} \geq -M * \left(1 - L_{pqrwij}\right) \qquad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V$$

Constraint (6) is modified to guarantee that person q is not matched with more persons than he can simultaneously offer a ride during each ride w of person q. The modified constraints (10) and (11) ensure that if person p rides with person q from i to j on ride r of p and ride w of q, then their departure times at node i have to be equal. If the time schedule of person p on ride r does not overlap with the time schedule of person q on ride w, their departure times at node i cannot be equal, and therefore constraints (10) and (11) causes L_{pqrwij} to be 0, meaning that person p cannot ride with person q from i to j on ride r of p and ride w of q.

With the introduced variables and constraints, as well as the modifications we make to basic ridematching constraints, and time and capacity constraints, we defined a complete ridematching model in this section. This ridematching model forms the second part of the matching algorithm we suggested in Section 4.2. In Section 4.4, we compare this matching algorithm with methods from literature. In Section 4.5, we provide an overview of the entire ridematching model.

4.4 Comparison With Existing Approaches

As indicated in Section 4.1, our suggested method differs from the existing ride-sharing methods. In this section, we make a short comparison between our method and existing methods. This should clarify the position of our method in relation to literature.

We relate our method to the methods of Drews & Luxen (2013) and Herbawi & Weber (2011). As discussed in Section 3.2, both methods deploy the technique of time-expanded graphs. In our matching algorithm, we do not use this technique. However, we use this time-expanded graph in this section to illustrate the difference between these methods and our matching algorithm.

Essentially, the methods of Drews & Luxen (2013) and Herbawi & Weber (2011) consist of two phases. In the first phase they set the time-expanded graph. Their first phase is quite similar to the first phase (shortest path computations) of our suggested method. In both cases, the first phase calculates, from a set of relevant stations, the stations that are (possibly) visited by an offered ride. In the case of Drews & Luxen (2013), the visited stations are the stations that can be visited within certain detour and delay boundaries, and in our case, these stations are the ones that are situated on the shortest path.

In the second phase, Drews & Luxen (2013) and Herbawi & Weber (2011) run an algorithm (Dijkstra) on the time-expanded graph to determine the rides a passenger should join in order to arrive at his destination with minimum delay. As mentioned in Section 4.2, the second phase of our method consists of the ridematching model, which we define in Section 4.3. So, what we have in common with methods from literature is that the second phase defines the carpool matches (which persons form a carpool). However, we use a different approach in matching persons for a carpool. The main difference is caused by the fact that Drews & Luxen (2013) and Herbawi & Weber (2011) assume a passenger's path is static, which means that he can only travel to another station by riding with someone else. Whereas we assume that each user can travel to another station by driving his own car, or by riding with someone else. The difference between these static and dynamic approaches can be explained with the illustration of a time-expanded graph, see Figure 4-15. Drews & Luxen (2013) and Herbawi & Weber (2011) make a clear distinction between passengers and ride offers. Only ride offers are added to the time-expanded graph and then they determine for each passenger the shortest path (path with smallest delay) from his origin to destination using these offered rides. This means that a passenger is matched with either ride 1 or ride 2 in Figure 4-15. Our method plans the rides of all users and then the matching model determines which rides show overlap. So, our model matches rides 1 and 2 in Figure 4-15, under the condition that all constraints, such as guaranteed arrival at destination, can be met.

As discussed in Section 3.2, the white nodes $v(1)$ represent an arrival event of person 1 at the corresponding station, and the grey nodes $u(1)$, represent a departure event of person 1 at the corresponding station. In Figure 4-15, we show how our method relates to other methods on a time-expanded graph. However, this time-expanded graph is only used to visualize the difference between the methods. We do not make use of a time-expanded graph in our ridematching model. In Section 4.5, we present the mathematical formulation of the complete ridematching model.

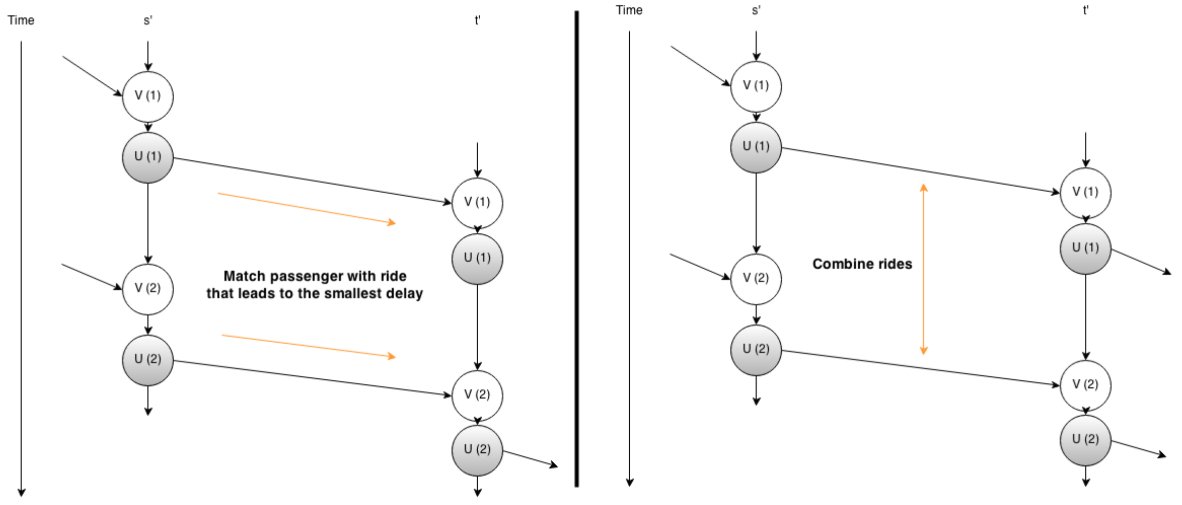


Figure 4-15: Time-expanded graph. Left graph indicates the second phase algorithm of Drews & Luxen (2013) and Herbawi & Weber (2011), right graph indicates our ridematching model.

4.5 Complete Ridematching Model

Based on the constraints we added (Sections 4.3.2.3 and 4.3.3.3), and the modifications we have made (Section 4.3.3.3) to the basic ridematching model, as well as the basic ridematching model itself (Section 4.3.1.3), we present the complete ridematching model on pages 44 and 45. All constraints in this complete ridematching model are discussed in previous Sections 4.3.1, 4.3.2, and 4.3.3. The objective function includes weight parameters for indicating the importance of each objective.

$$Minimize \left(\begin{array}{c} w_1 * \sum_{p\in P}\sum_{r\in R}\sum_{i,j\in V} t_{ij} * Z_{prij} + w_2 * \sum_{p,q\in P}\sum_{r\in R}\sum_{i\in V} S_{pqri} + w_3 * \sum_{p\in P}\sum_{r\in R}\left(l_p - D_{p,r,des_{pr}}\right) + \\ w_4 * \sum_{p\in P}\sum_{r\in R}\left(\left(A_{p,r,des_{pr}} - D_{p,r,or_{pr}}\right) - \sum_{i,j\in V} t_{ij} * x_{prij}\right) \end{array} \right)$$

s.t.

(1) $Y_{pprij} = 0$ $\quad \forall p \in P, \forall r \in R, \forall i,j \in V$

(2) $\sum_{q\in P} Y_{pqrij} + Z_{prij} = x_{prij}$ $\quad \forall p \in P, \forall r \in R, \forall i,j \in V$

(3) $\sum_{q\in P}\sum_{i\in V} Y_{pqrij} + \sum_{i\in V} Z_{prji} \leq 1 + C_{p,r-1,j}$ $\quad \forall p \in P, \forall r \in R, \forall j \in V$

(4) $Y_{pqrij} \leq \sum_{w\in R} Z_{qwij}$ $\quad \forall p,q \in P, \forall r \in R, \forall i,j \in V$

(5) $S_{pqrj} \geq \sum_{i\in V} Y_{pqrij} - \sum_{i\in V} Y_{pqrji}$ $\quad \forall p,q \in P, \forall r \in R, \forall j \in V$

(6) $\sum_{p\in P}\sum_{r\in R} L_{pqrwij} \leq Cap_q$ $\quad \forall q \in P, \forall w \in R, \forall i,j \in V$

(7) $A_{prj} = D_{pri} + t_{ij}$ $\quad \forall p \in P, \forall r \in R, \forall i,j \in V$

(8) $D_{pri} \geq A_{pri}$ $\quad \forall p \in P, \forall r \in R, \forall i \in V$

(9) $A_{p,r,des_{pr}} \leq l_{pr}$ $\quad \forall p \in P, \forall r \in R$

(10) $D_{p,r,or_p} \geq e_{pr}$ $\quad \forall p \in P, \forall r \in R$

(11) $D_{pri} - D_{qwi} \leq M * \left(1 - L_{pqrwij}\right)$ $\quad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V$

(12) $D_{pri} - D_{qiw} \geq -M * \left(1 - L_{pqrwij}\right)$ $\quad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V$

(13) $\left(A_{p,r,des_{pr}} - D_{p,r,or_{pr}}\right) - \sum_{i,j\in V} t_{ij} * x_{prij} \leq m_{pr}$ $\quad \forall p \in P, \forall r \in R$

(14) $C_{pri} \leq 1 - \sum_{j\in V} Z_{prij}$ $\quad \forall p \in P, \forall r \in R, \forall i \in V$

(15) $C_{pri} \geq \sum_{j\in V} Z_{prji} - \sum_{j\in V} Z_{prij}$ $\quad \forall p \in P, \forall r \in R, \forall i \in V$

(16) $C_{pri} \geq C_{p,r-1,i} - \sum_{j\in V} Z_{prij}$ $\quad \forall p \in P, \forall r \in R, \forall i \in V$

(17) $C_{pri} \leq \sum_{j\in V} Z_{prji} + C_{p,r-1,i}$ $\quad \forall p \in P, \forall r \in R, \forall i \in V$

$$\sum_{w>r}\sum_{j\in V} Z_{pwij} \geq C_{pri} \qquad \forall p \in P, \forall r \in R, \forall i \in V \tag{18}$$

$$\sum_{q\in P}\sum_{i\in V} Y_{p,q,r,i,des_{pr}} + \sum_{i\in V} Z_{p,r+1,des_{pr},i} \leq 1 + C_{p,r,des_{pr}} \qquad \forall p \in P, \forall r \in R \tag{19}$$

$$L_{pqrwij} \leq Y_{pqrij} \qquad \forall p,q \in P, \forall r,w \in R, \forall\, i,j \in V \tag{20}$$

$$L_{pqrwij} \leq Z_{qwij} \qquad \forall p,q \in P, \forall r,w \in R, \forall\, i,j \in V \tag{21}$$

$$\sum_{w\in R} L_{pqrwij} = Y_{pqrij} \qquad \forall p,q \in P, \forall r \in R, \forall i,j \in V \tag{22}$$

$$Y_{pqrij},\, Z_{prij},\, S_{pqri}, C_{pri}, L_{pqrwij} \in \{0,1\}, D_{pri,} A_{pri} \geq 0 \qquad \forall p,q \in P, \forall r,w \in R, \forall i,j \in V \tag{23}$$

Decision variables

- Y_{pqrij}, binary variable indicating whether person p rides with person q from i to j on ride r. So $Y_{pqrij} = 1$ when person p sits in the passenger's seat and person q sits in the driver's seat from i to j on ride r, and $Y_{pqrij} = 0$ otherwise.
- Z_{prij}, a binary variable indicating whether person p drives from i to j on ride r. So $Z_{prij} = 1$ when person p sits in the driver's seat from i to j on ride r, and $Z_{prij} = 0$ otherwise.
- D_{pri}, a continuous variable indicating the departure time of person p at node i on ride r.
- A_{pri}, a continuous variable indicating the arrival time of person p at node i on ride r.

Dependent variables

- S_{pqri}, a binary variable indicating the starting point of the carpool on a certain, in case person p carpools with person q during this ride. $S_{pqri} = 1$ when person p meets person q for a carpool at node i on ride r, and $S_{pqri} = 0$ otherwise.
- C_{pri}, a binary variable indicating the node i where the car of person p is parked on ride r. $C_{pri} = 1$ when the car of person p is parked on node i on ride r, and $C_{pri} = 0$ otherwise.
- L_{pqrwij}, binary variable indicating whether person p rides with person q from i to j on ride r of p and ride w of q. So $L_{pqrwij} = 1$ when, on ride r of person p, person p sits in the passenger's seat and, on ride w of q, person q sits in the driver's seat from i to j, and $L_{pqrwij} = 0$ otherwise.

Parameters

- $w_{\propto}$, a weight parameter to specify the importance of each objective, with $\alpha = \{1,2,3,4\}$, indicating the number of the objective.
- x_{prij}, a binary parameter indicating which edges are part of a person's shortest path on a certain ride. So $x_{prij} = 1$ when edge (i,j) is part of the shortest path of person p on ride r.
- Cap_p, a parameter indicating how much passengers person p can take simultaneously.
- l_{pr}, a time parameter indicating the latest arrival time at the destination of ride r of person p.
- e_{pr}, a time parameter indicating the earliest departure time at the origin of ride r of person p.
- m_{pr}, a time parameter indicating the maximum waiting time during ride r for person p.
- M, a parameter which is sufficiently large to cover the difference in arrival times at node i of persons p and q on ride r.

4.6 Conclusion

In this chapter, we introduced a two-stage method for suggesting carpools. The first part of the method consists of shortest path computations. We assume shortest path techniques are known and consider these computations as ‘black box’ computations in this chapter. The shortest paths are input to the second stage of the method, the ridematching model. We explained and mathematically formulated the model’s functionalities in three stages, which are:

- *Basic Ridematching Model*
  The model filters overlap in routes and sets up carpools between persons with overlapping routes when the passenger is able to arrive at his destination with the set of offered rides. Passenger and driver can drive to a meeting point to start the carpool, and the passenger can hop between carpools.
- *Time & Capacity Extension*
  Besides the basic functionalities, the model filters overlap in time schedules and sets up carpools between persons with overlapping time schedules when the driver is able to pick up the passenger (capacity).
- *Return Restrictions*
  Besides the basic, time, and capacity functionalities, the model sets up carpools between persons, when each person is able to reach all his destinations in a certain time period and, when the car is parked at a meeting point, is able to return to his car on a later ride. Due to the return restrictions, the model has to cope with multiple rides per person in a certain time period.

5. Numerical Experiments

This chapter answers the following research question.

- *Which benefits can be realized with the developed carpooling method?*

We use the rides driven by Significant personnel to evaluate the performance of the carpooling method. In Section 5.1, we discuss the characteristics of the data collected of Significant-related rides and the determination of shortest paths. In Section 5.2, we initialize the model's parameters and settings. Subsequently, we perform experiments on the ridematching model and discuss the model's performance in Section 5.3. Section 5.4 compares the functionalities of the model with that of existing apps. Finally, Section 5.5 summarizes this chapter.

5.1 Model Input

In a nutshell, the ridematching model turns input, such as shortest paths, person characteristics, and time schedules into packages of matched rides, see Section 4.2. In this section, we discuss the input data we prepare to perform experiments on the ridematching model. Section 5.1.1 describes the characteristics of the collected data on rides driven by Significant personnel. Subsequently, Section 5.1.2 discusses the computations of the shortest paths and the determination of edges on shortest paths.

5.1.1 Data Collection of Significant's Rides

As discussed in Section 2.1, we collected data of Significant-related rides of all employees during four weeks. These trips are used to evaluate the performance of the ridematching model. During a four-week period, we collected a total of 1416 one-way rides made by Significant employees. Since we do not take weekends into account, each week consists of five days. So, these 1416 rides are conducted in 20 days. Figure 5-1 shows the amount of rides driven on each of these 20 days.

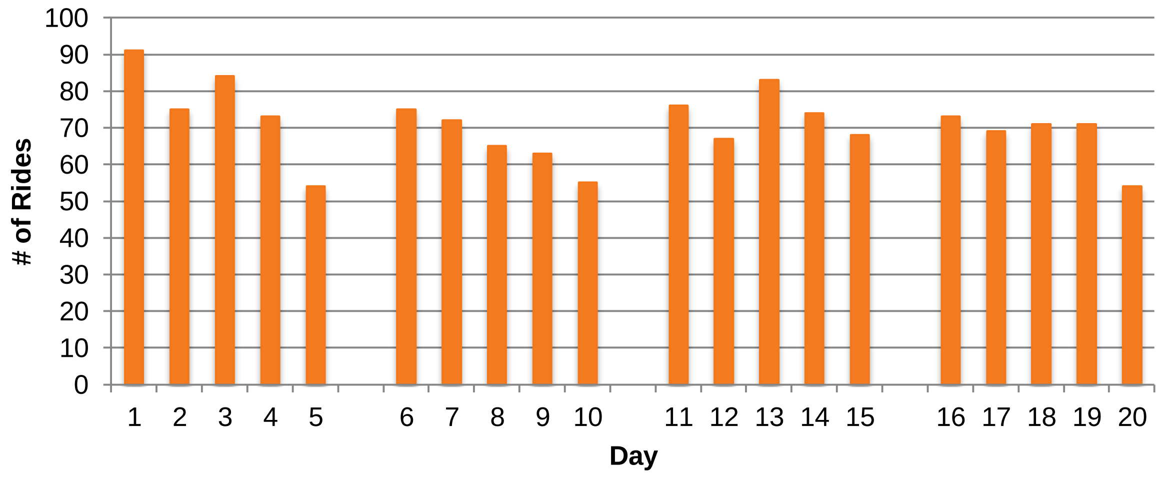


Figure 5-1: Distribution of the collected rides over the days.

Because employees start and finish their working days at home, a main part of the collected rides consists of rides to or from a home address. This part makes up 87% of the 1416 rides. The other 13% consists of rides from client to office, from office to client, or from client to client. Figure 5-2 shows that most of the rides were driven in the early morning or late afternoon. This distribution of rides matches with the high percentage of rides to or from a home address, because employees generally drive from home to work in the early morning and drive from work to home in the late afternoon.

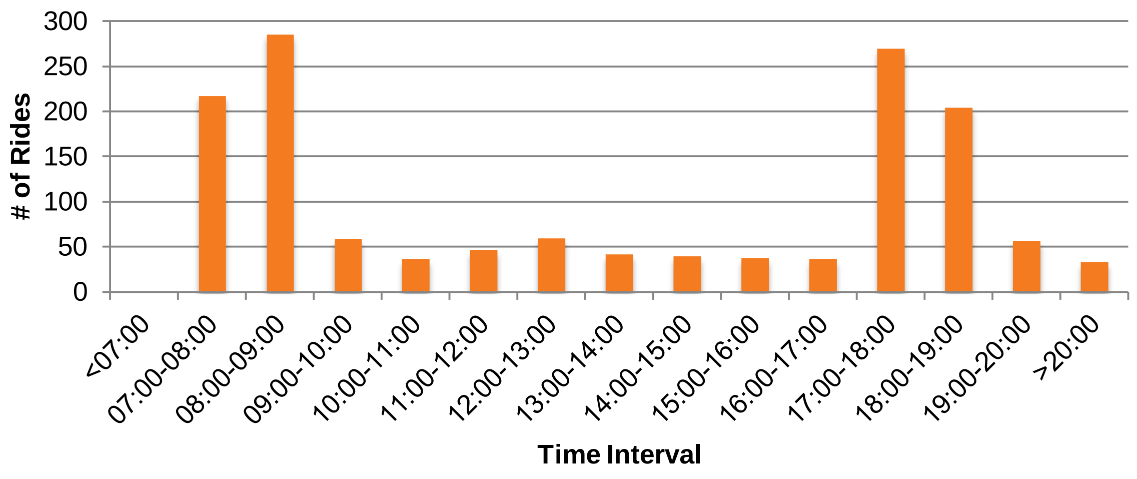


Figure 5-2: The distribution of the rides over the time intervals

The employees of Significant were asked to keep track of their rides by filling out prepared forms. On these forms, employees could fill in their destinations of each day, and in which hour of the day they arrived at this destination. Because each employee starts the day at home, a complete overview of the rides driven by the employees, including corresponding time schedules, can be generated. See Table 5-1 for an example of the data filled in by one employee on a certain day. Based on the data in the table, we can conclude that this employee drove from his home address to Utrecht and arrived between 7 and 8 am in Utrecht. Consequently he drove from Utrecht to Barneveld and arrived there between 1 and 2 pm, etcetera. Because we have access to the employees' addresses, we replaced 'home' by the employee's hometown. In total, we collected 78 different destinations, including employees' hometowns, see Figure 2-1 for an overview of these destinations (in relation to their visiting frequency).

Destination	Period
Utrecht	07:00-08:00
Barneveld	13:00-14:00
Home	19:00-20:00

Table 5-1: Example of data delivered by an employee on a certain day

In conclusion, the rides derived from the data collection have two main characteristics:

- The rides are defined on city level. Rides occur between an origin city and a destination city, due to the inability to retrace exact addresses of visited locations.
- Time schedules of the rides are defined on an hourly level. Because employees could not retrieve exact arrival times at their destinations, they indicated the arrival time in a one-hour interval.

In the data collection, also some rides within the same city were gathered, for example, a ride from Utrecht to Utrecht. Naturally, these rides are not serviceable in our experiments, and therefore we delete these rides from the set of rides we use for experiments on the ridematching model. In Section 5.1.2, we elaborate on the determination of the shortest path between two cities.

5.1.2 Shortest Path Computations

In Section 4.2, we made a clear distinction between shortest path computations and the ridematching model. The shortest paths serve as input for the ridematching model. We stated that we consider the shortest path computations as a ‘black box’. Furthermore, we assumed that output of this ‘black box’ consists of output variables x_{pij}, indicating whether the edge (i, j) is on the shortest path of person $p \epsilon P$ (with $x_{pij} = 1$ indicating edge (i, j) is part of the shortest path of person p and $x_{pij} = 0$ otherwise). Where the vertices (i and j) represent relevant carpool places, which are starting points and destinations of the users’ trips and possible meeting points for a carpool, such as Park and Ride’s. In this section, we ‘open the black box’ by explaining how we define the edges on the shortest path of a person, used for experiments on the ridematching model.

As discussed in Section 5.1.1, we collected data of rides on city level. With information on a person’s origin city and destination city, we have to determine the edges that make up the shortest path between these cities. As already mentioned in this section, these edges are connections between relevant carpool places. The first part of the relevant carpool places we use for our shortest path computations, consists of the 78 destinations from the data collection, which represent starting points and destinations of the employees’ trips. The second part of the relevant carpool places consists of possible meeting points for a carpool. For this, we retrieved a list from Carpoolplein.nl with the names and coordinates of 358 carpool parking lots in the Netherlands. So, the network we use for shortest path computations consists of 436 nodes in total. The goal of shortest path computations is, given an origin city and destination city, to determine which of these 436 nodes are located on the path between these cities, and in which order, see Figure 5-3.

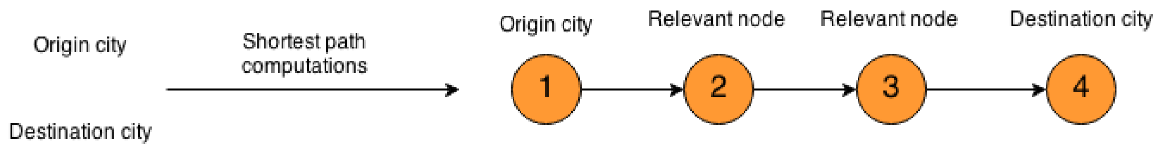


Figure 5-3: Overview of input and output of shortest path computations.

Besides knowing which nodes are located in which order, on a path between two cities, we also need to know the travel time between two nodes. As stated in Section 4.2, we associate each edge (i, j) with travel time t_{ij}, and assume a shortest path is the path of subsequent edges from origin to destination that takes the least amount of time. Thus, in order to perform the shortest path computations, we need travel times between all 436 nodes (which is the same as the travel times of all possible edges). Following the statement of Drews & Luxen (2013), who opt for a pre-determined table of travel times (see Section 3.2), we construct a $436 * 436$ matrix containing travel times between all relevant nodes. This means that we have to retrieve 189660 $(= 436 * 436 - 436)$ travel times. To collect these travel times, we make use of Google Maps API and Bing Maps API. An API is “a programming language that allows two different applications to communicate, or interface, with each other” (Gunelius, 2011). Services, such as Google and Bing, release their API so that external software developers can build new products powered by their maps services. In our case, we communicate with Google Maps API and Bing Maps API via Microsoft Excel. With Excel we are able to program an automatic process of retrieving the travel time of each possible node-to-node combination in the $436 * 436$ matrix. This process can be summarized in two steps:

1. Retrieve coordinates of all 78 destinations of Significant’s ride collection by communicating with Google Maps API. This provides us with coordinates of the centre of each city. The coordinates of the carpool parking lots are already known.
2. For each possible combination in the $436 * 436$ matrix, communicate the corresponding coordinates with Bing Maps API and retrieve the travel time.

The reason for using two kinds of API (Google Maps and Bing Maps) is that Google Maps API provides accurate coordinates of the centre of a city based on the city name, while Bing Maps API has more trouble providing accurate centre coordinates. However, we use the Bing Maps API to retrieve travel times, because the Google Maps API restricts communication on travel times to 2500 requests per day. We have to perform 189660 requests, and the Bing Maps API does not impose restrictions on the daily amount of requests.

With the travel times between all 436 nodes, we determine shortest paths of the rides in the data collection. The consideration whether a node is on the shortest path is based on two assumptions. These assumptions lead to some drawbacks, which we discuss later in this section. The two assumptions are:

- Every carpool parking lot within a 5 minute reach of the travel time between origin and destination is on the shortest path. With a 5 minute reach we mean that the travel time from origin to the carpool parking lot plus the travel time from the carpool parking lot to destination, is smaller than the travel time from origin to destination plus 5 minutes. Because the carpool parking lots are located near highway exits or access roads, these places are reached quite easily. So when a carpool parking lot is located near an exit and access road of a highway on the shortest path, it will not take more than 5 minutes to visit the parking lot.
- Every city within a 10 minute reach of the travel time between origin and destination is on the shortest path. Because we calculate from the centre of a city, cities are harder to reach than a parking lot along the highway. Therefore, we determine that a city is located on the shortest path when it takes no more than 10 minutes to visit the centre of the city.

To explain the process of defining the shortest path between two cities, we describe an example of a ride between Utrecht and Barneveld. After a search in the pre-determined timetable, we conclude that 8 nodes satisfy the shortest path assumptions, of which 3 city nodes and 5 carpool parking nodes. The origin city and destination city make up 2 of the 3 city nodes. In Figure 5-4**,** we show the 8 nodes of the shortest path on a map. The figure shows that all 8 nodes are located near the route from Utrecht to Barneveld.

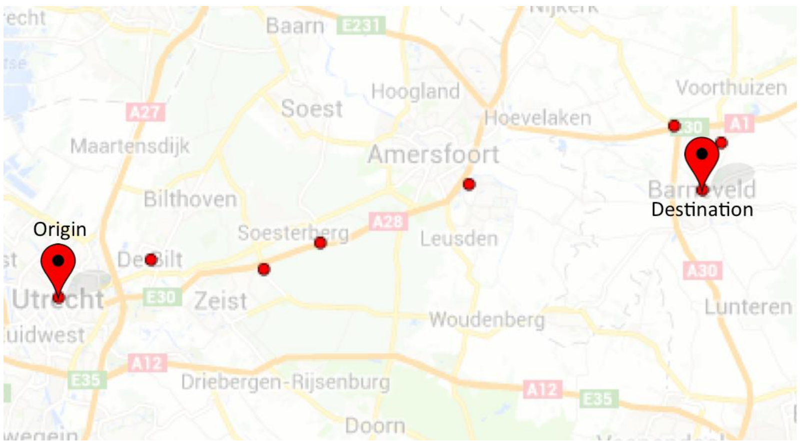


Figure 5-4: Nodes of shortest path between Utrecht and Barneveld (Google).

The visiting order of the nodes on the shortest path is determined by sorting the nodes on the travel time between the origin and the node. We assume that the node on the shortest path with the smallest travel time from origin to this node, is the first visited node on the shortest path, and the node with the largest travel time from origin to this node, is the last visited node. To illustrate this, we order the 8 nodes of Figure 5-4**,** based on the travel time between origin (Utrecht) and each node. The edges between subsequent nodes and corresponding travel times are input to the ridematching model, see Table 5-2.

Edge	**From**	**To**	**Time (min)**
1	Utrecht	De Bilt	13.6
2	De Bilt	P Carpool Zeist/ Den Dolder A28	11.6
3	P Carpool Zeist/ Den Dolder A28	P Carpool Soesterberg A28	2.8
4	P Carpool Soesterberg A28	P Carpool Leusden A28	6.9
5	P Carpool Leusden A28	P Carpool Transferium Barneveld-Noord (A1)	13.0
6	P Carpool Transferium Barneveld-Noord (A1)	P Carpool Barneveld A30	6.7
7	P Carpool Barneveld A30	Barneveld	8.3
			62.9

Table 5-2: Input to the ridematching model.

In this section, we discussed the method of determining shortest paths, used for experiments on the ridematching model. In Section 4.2, we made a clear distinction between shortest path computations and the ridematching model. The shortest paths serve as input for the ridematching model. The drawback of the method we use for shortest path computations, is that it is based on assumptions. We assume every carpool parking lot within a maximum range of 5 minutes of shortest path duration, is located on the shortest path. Similarly, every city within a 10 minute range is adopted in the shortest path. Furthermore, we assume that subsequent nodes, ordered on travel times between the origin and each node of the shortest path, provide a good representation of the edges of the shortest path. We mention three effects of this assumption based approach.

- In the ridematching model, we define the total travel time from an origin to a destination as the sum of the travel times of the edges on the shortest path between origin and destination. However, looking at the example of a trip between Utrecht and Barneveld, the sum of the travel times of the edges is 63 minutes, while direct travel time between Utrecht and Barneveld is 35 minutes. So, in our experiments travel times between origin and destination are larger than in real life. This is caused by the fact that nodes of the shortest path, are not exactly on the route from origin to destination, but are based on assumptions (within 5 or 10 minutes of shortest path duration). So, the travel time of an edge between any two of these nodes also includes the time buffer (<5 minutes for a carpool parking lot and <10 minutes for a city) of both nodes.
- Sorting the nodes on travel time between the origin and the node, does not always generate logical edges. We show the edges of Table 5-2 on a map in Figure 5-5. Edge 5, 6, and 7 do not represent a realistic situation. The end of edge 5 is *P Carpool Transferium Barneveld-Noord (A1)* and the end of edge 6 is *P Carpool Barneveld A30* (Table 5-2). In a realistic situation, we would first drive to *P Carpool Barneveld A30* and then to *P Carpool Transferium Barneveld-Noord (A1)*. So, in our shortest path computations, it can occur that edges do not represent a real life situation. Nodes that are located close to each other may cause non-realistic edges, because one of these nodes may be further away from the shortest path (5 minutes), than the other (0.5 minutes). This may lead to the situation of Figure 5-5.
- When a ride from a to b has two (or more) alternative routes with comparable travel times, our method will appoint nodes along both routes to the shortest path from a to b. This leads to edges that connect

nodes from one route with nodes from the alternative route, but in real life a person does not drive both routes.

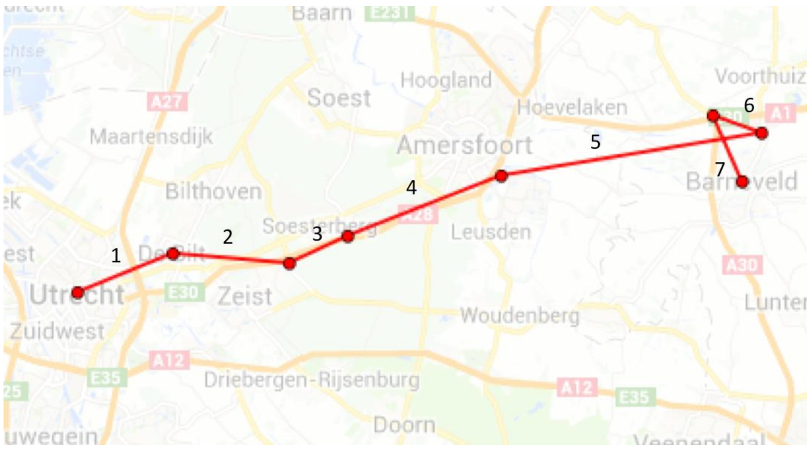


Figure 5-5: Edges of the shortest path between Utrecht and Barneveld (Google).

Due to the assumptions we make, shortest paths and travel times of routes may not fully represent a real life situation. However, because we apply the same shortest path computations to every ride, we provide consistent input to the ridematching model, enabling the model to generate representative carpool matches. In Figure 5-6, we show the edges of the shortest path between Zeist and Barneveld. Edge 3, 4, 5, 6, and 7 of the shortest path between Utrecht and Barneveld (Figure 5-5), are also included in the shortest path between Zeist and Barneveld, illustrating consistency of the shortest path computations.

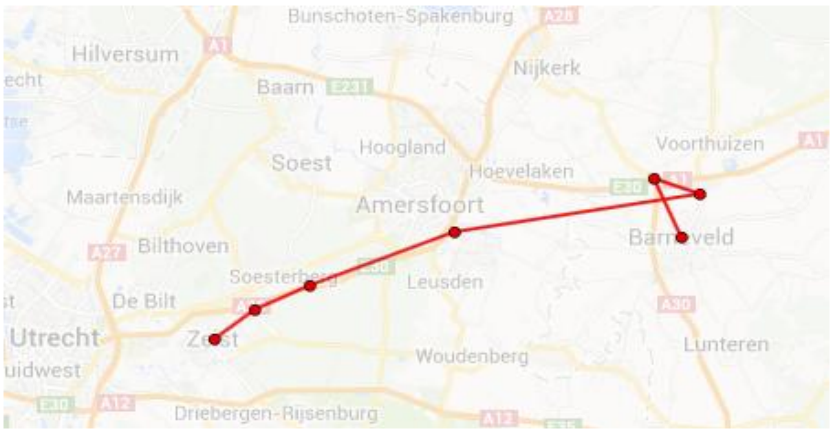


Figure 5-6: Edges of the shortest path between Zeist and Barneveld (Google).

The drawback of the alternative routes with comparable travel times can lead to unrealistic carpool matches. Unrealistic carpool matches occur when the model suggests that person i rides with person j on edge (a, b), where node a is on one of the alternative routes of person j, and node b on the other alternative route of person j, while edge (a, b) does represent a real life shortest path edge of person i. This means that the model suggests that person j offers a ride to person i on an edge that j does not drive in a real life situation. Figure 5-7 shows the edges determined by our method for a shortest path between Hellevoetsluis and Arnhem. Because two different routes exist from Hellevoetsluis to Arnhem with comparable travel times, the method appoints nodes along both routes and draws edges between nodes on both routes. Another unrealistic carpool match occurs when the model suggests that persons i and k ride with person j, and that person i needs to be picked up along one of the alternative routes of person j, and person k needs to be picked up along the other alternative route of person j.

So due to the shortest path assumptions, we have to manually check whether the carpool suggestions do represent a real life situation in our experiments. Since the shortest paths serve as input to the ridematching model, the drawbacks of the method to determine these shortest paths does not affect the validity of the ridematching model. The only consequence of incorrect input to the model is incorrect output. This means that the model can suggest unrealistic carpools, when the given shortest paths do not consist of real life edges. In Section 5.3, we examine experiment results when we exclude the possibility of unrealistic carpool suggestions.

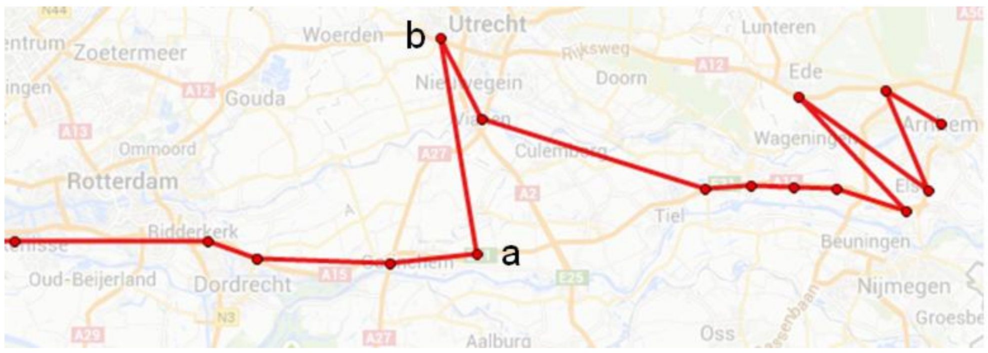


Figure 5-7: Edges determined by our shortest path computations from Hellevoetsluis to Arnhem (Google).

5.2 Parameter Initialization & Model Settings

In Section 4.3, we mentioned that weights have to be assigned to each objective of the ridematching model. These weights have to be determined such that the model generates desired outcomes. In this section, we initialize the weights of the model. Furthermore, we elaborate on the settings of the ridematching model we perform experiments on.

We programmed the ridematching model (Section 4.5) in the optimization modelling software of AIMMS. Validation of the model is done by using various sets of rides from the data collection. During the building process of the model, as well as afterwards, we manually checked whether outputs of the model correspond with outputs that the model should give according to the defined functionalities. As far as we know, the model functions according to the functionalities as described in Chapter 4.

One characteristic of the rides we gathered amongst Significant personnel is the definition of time schedules on an hourly level, see Section 5.1.1. Employees indicated in which hour they arrived at their destination, for example between 7 and 8 am. The ridematching model we defined in Section 4.3 requires the inputs earliest departure time at the origin city and latest arrival time at the destination city. Because we only collected arrival time intervals of employees, we make some assumptions. We consider the upper bound of a time interval to be the latest arrival time, and the lower bound of a time interval to be the earliest arrival time. For example, an employee living in Amsterdam who indicated that he arrived between 7 and 8 am at Utrecht. If this is his first ride of the day, he drives from Amsterdam to Utrecht, which takes 70 minutes (= 1 hour and 10 minutes) with our method of shortest path calculation (see Section 5.1.2). We can calculate the earliest departure time by $07{:}00 - 1{:}10 = 5{:}90$. In this case, (time) input to the ridematching model is:

- Earliest departure time $= 5{:}90$
- Latest arrival time $= 08{:}00$

This means this person could have been matched with a colleague that indicated to arrive at Utrecht between 6 and 7 am, or a colleague that arrived at Utrecht between 8 and 9 am.

Other parameters to the model are maximum waiting time and maximum capacity, see Section 4.3. An important objective of the ridematching model is to reduce waiting times incurred during the ride to a minimum, as any waiting time is undesirable for each employee. However when waiting times do occur, an employee cannot incur more waiting time during the ride than a certain maximum. In Section 4.3.2.1, we stated that "*with waiting time we mean the difference between the actual trip duration and the fastest trip duration. So, we only consider waiting times at intermediate nodes between starting point and destination as waiting time. A deviation from earliest departure time at the starting point or from the latest arrival time at the destination is not considered as waiting time*". In our experiments we set the maximum waiting time to 10 minutes, and maximum total deviation from latest arrival time at the destination to 60 minutes (size of time interval). Besides this, we assume each employee can take at most 3 passengers simultaneously when driving, so the maximum capacity is 3.

Parameter	**Value**
Earliest departure time	Lower bound arrival time interval – travel time
Latest departure time	Upper bound arrival time interval
Maximum waiting time	10 minutes
Maximum capacity (maximum passengers)	3

Table 5-3: Parameter settings.

Table 5-3 summarizes the parameter settings we use in our experiments, in which travel time equals the sum of the travel times of all edges of the shortest path. With these settings we determine the value of the weights we assign to each objective of the ridematching model. The four objectives of the ridematching model (Section 4.3.3.2) are to minimize:

1. Total time spent as a driver by all persons
2. Deviation of arrival time at destination from latest arrival time
3. Total waiting time of all persons
4. The number of carpools

In the experiments we allocate weight 1 to objective 1 of minimizing the time driven by all persons, and weight 2 to objective 2, etcetera. To determine which weight values generate the most desirable model performance, we performed experiments on day 1 of the data collection. This day contains the most rides, compared to all other days of the data collection, see Section 5.1.1. We initialize the weights of the model's objectives by running the model several times for this day, and change the weights with each run (without changing any other settings). We assess the model's performance on 4 indicators. Each indicator represents one of the model's objectives. The carpooled time is an indicator of objective 1, minimizing the time driven of all persons. The total deviation from latest arrival times is an indicator of objective 2, the total waiting time represents objective 3, and the number of carpools matches the performance of objective 4.

First, we establish the optimal value of the carpooled time by giving the objective 1, a weight of 1, and the other objectives a weight of 0. Figure 5-8 shows the results of this experiment. Experiment 1 indicates that the carpool optimum is 1528 minutes. This time is reached when we only optimize the model on the objective of minimizing the time spent as a driver by all persons. However, the other objectives cannot be ignored. Solely minimizing on objectives 2, 3, or 4 will lead to respectively zero deviation from latest arrival times, zero total waiting time, or zero carpools. Our goal is to assign weights to the objectives such that the model provides a solution (almost) equal to the carpool optimum, and that requires a minimum total deviation from latest arrival times, a minimum total waiting time and a minimum amount of different carpools.

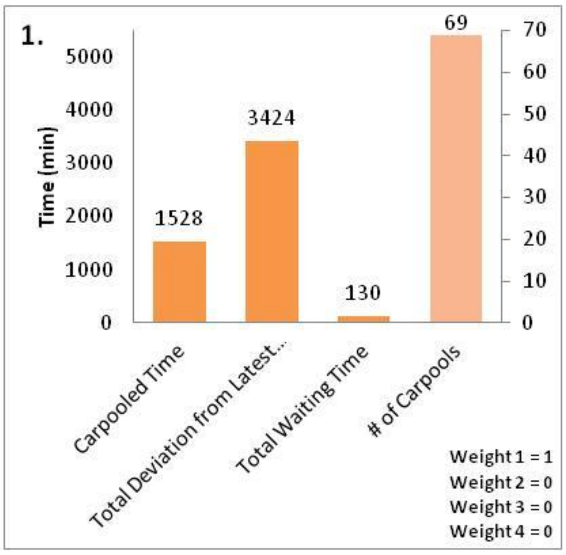


Figure 5-8: Outcomes of the ridematching model, when only the objective of minimizing total time spent as a driver, is optimized.

To avoid that the carpool time will strongly deviate from the optimum level, we conduct an experiment where the objective of minimizing time spent as a driver by all persons is superior to the other objectives (see Figure 5-9). Experiment 2 shows that assigning relatively small weights to objectives 2, 3, and 4 leads to a solution with an optimum carpool time, while the indicators for the other objectives show better results compared to the situation where the other objectives are ignored (experiment 1). However, in Section 4.3.3.2, we stated that the goals of minimizing the number of carpools and minimizing deviation from latest arrival times (objectives 2 and 4) are inferior to the main goals of maximizing the total carpooled time and minimizing the total waiting time (objectives 1 and 3). So objective 3 should be assigned a bigger weight. In experiment 3 (Figure 5-9), we assign equal weights to these superior objectives, and relatively small weights to objectives 2 and 4. With this weight configuration, the carpooled time is still 1528 minutes, and the total waiting time reaches the minimum value of 0 minutes. We would expect an increase in total deviation from latest arrival time to compensate for the zero waiting time. Remarkably, the model also returns better scores on objectives 2 and 4.

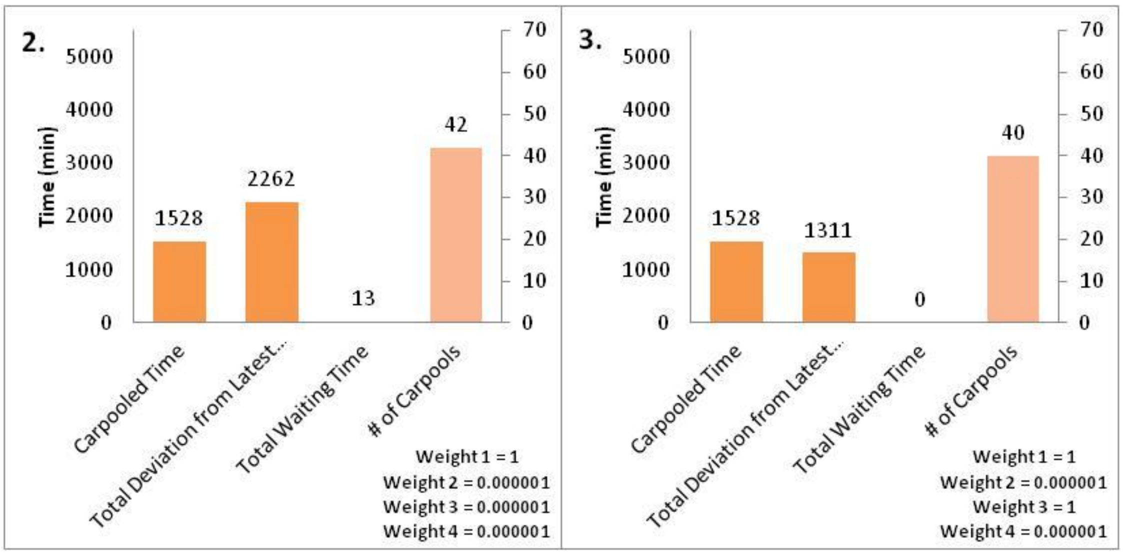


Figure 5-9: Experiment 5 where objective 1 is superior to the other objectives, and experiment 6 where objectives 1 and 3 are superior to the other objectives.

In experiment 3, we used a weight configuration that leads to optimal values on the indicators for objectives 1 and 3. To see whether we are able to make more gains on the total deviation from latest arrival times (objective 2), we assign more weight to this objective in experiment 4 (Figure 5-10), while keeping weights of the other objectives at the same level as in experiment 3. With this adjustment to the weights, indicators of objectives 1 and 3 still have optimal values, and we gain 24 minutes on the total deviation from latest arrival times.

In experiments 3 and 4, we assign equal weights to objectives 1 and 3. However, minimizing total waiting time (objective 3) should be assigned more weight than minimizing total time spent as a driver by all persons (objective 1). Because the time that a person waits for a carpool, should be in proportion to the duration of the carpool he enables by waiting at a meeting point. For example, in case of experiment 4 (Figure 5-10), we assign a value of 1 to weight 1 and a value of 1 to weight 3. With this configuration a person may wait up to 10 minutes to make a carpool of 10 minutes possible. It is unlikely that a person is willing to wait 10 minutes for a carpool of 10 minutes, so in experiment 8 we assign a value of 0.5 to weight 1 and a value of 1 to weight 3. This way, a person waits at most 10 minutes to enable a carpool of 20 minutes. Although the configurations in experiments 4 and 5 lead to the same results, we consider the waiting time twice as important as the carpooled time, which is the case in experiment 5.

Based on the experiments concerning the determination of the weights of the model's objectives, we use the configurations of experiment 5, Figure 5-10, for further experiments on the model. We did not perform extensive experiments to determine these weights, but reasoned why these weights are satisfactory for our experiments on the ridematching model. The ridematching model has two main objectives (1 and 3), so we assigned sufficiently large values to these objectives. The two inferior objectives get relatively small weights. We argue that small changes in these weights will not lead to significant improvements in the model's performance. Furthermore, we showed that these weights lead to optimal values for the main objectives 1 and 3, and acceptable values for objectives 2 and 4, for day 1 of our data collection.

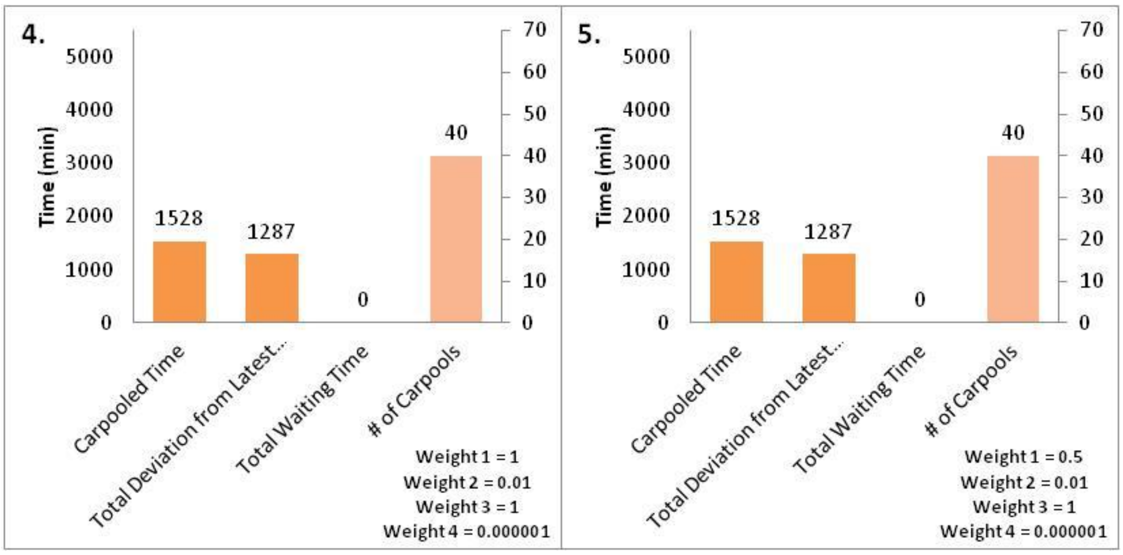


Figure 5-10: In experiment 7 we try to reduce total deviation from latest arrival times, while maintaining optimal values for objectives 1 and 3, experiment 8 adjusts the weight of objective 1 in proportion to objective 3.

In the beginning of this section, we considered the upper bound of a time interval to be the latest arrival time, and the lower bound of a time interval to be the earliest arrival time. So, when an employee indicated that he arrived between 7 and 8 am at Utrecht, then he could have been matched with a colleague that indicated to arrive at Utrecht between 6 and 7 am, or a colleague that arrived at Utrecht between 8 and 9 am. However, to validate the model, we adjust the settings such that an employee can only be matched with a colleague with an equal arrival time interval, so for example, earliest arrival time is 07:01 and latest arrival time is 08:00. Consequently, this employee cannot be matched with a colleague with an earlier or later arrival time interval. We expect to see a decrease in carpooled time, because we tighten the time schedules of the employees. The results in Figure 5-11 endorse this expectation. Compared to experiment 5 of Figure 5-10, tightening the time schedules leads to a reduction of 37% in carpooled time. Remarkably, the total waiting time and the number of carpools are equal to the situation of experiment 5. A reason for zero total waiting time is that no extra carpools can be made when incorporating waiting times. A reason for the equal number of carpools is that the model matches more persons for small distance carpools in experiment 6.

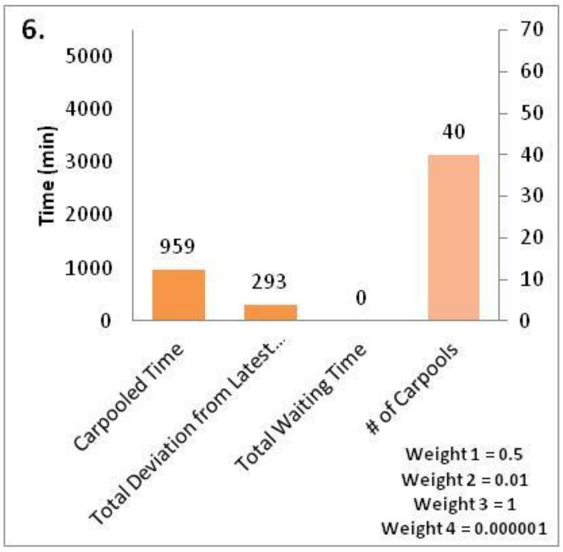


Figure 5-11: Ridematching results when difference between latest and earliest times is 59 minutes.

For further experiments we make use of the 'old' time schedule settings, so an employee can be matched with a colleague within the previous or subsequent time arrival time interval. In Table 5-4, we summarize the all parameter settings used in further experiments, which we conduct in Section 5.3.

Parameter	**Value**
Earliest departure time	Lower bound arrival time interval – travel time
Latest departure time	Upper bound arrival time interval
Maximum waiting time	10 minutes
Maximum capacity (maximum passengers)	3
Weight 1	0.5
Weight 2	0.01
Weight 3	1
Weight 4	0.000001

Table 5-4: Initialized parameter data.

5.3 Experiments

In this section, we show the results of applying the ridematching model to the rides driven by Significant personnel in a period of four weeks. We run the ridematching model 20 times, once for each day.

Figure 5-12 shows the main results of experiments with the ridematching model on the rides driven by Significant employees. This figure indicates what part of all routes employees could be riding with a colleague. For example, consider an employee a that has a ride duration of 100 minutes, and an employee b that has a ride duration of 50 minutes, and they form a carpool of 30 minutes. Then the percentage of all rides that the employees do not drive (or ride with a colleague) is $\frac{30}{100+50} = 20\%$. We emphasize that this percentage cannot reach 100%, as somebody has to drive when two (or more) persons are carpooling. In our example, the maximum percentage is $\frac{50}{100+50} = 33\%$. In Section 4.2, we stated that we assume a linear relationship between travel time and travelling distance. Which means that Figure 5-12 shows what percentage of the total distance driven could have been saved by carpooling, which we call the carpool potential.

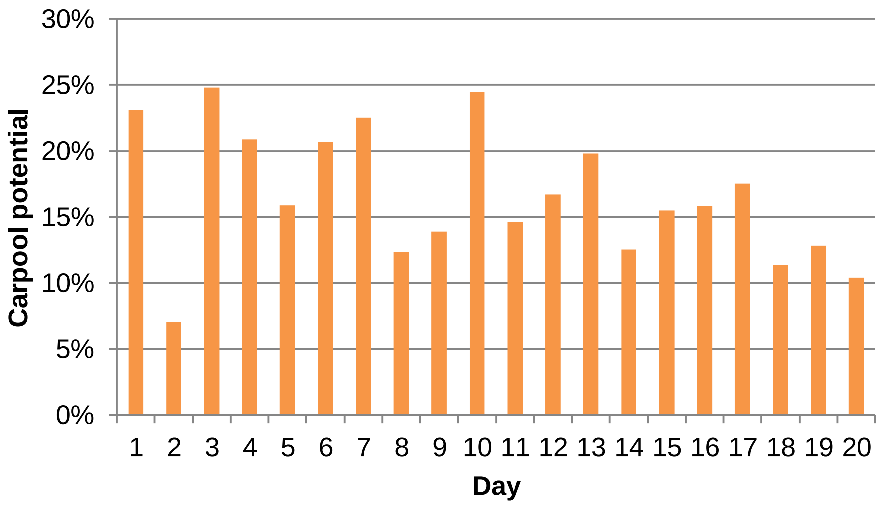


Figure 5-12: Percentage of the total travel time of all rides that employees do not drive.

Based on our experiments, Significant employees could have saved 7% (day 2) to 25% (day 3) of the distance driven. Interestingly, the model did not assign any waiting time at each of the 20 days. In total, of the 1416 rides, the model suggest that persons join the ride of a colleague on 511 rides. We state that the model suggests 511 carpools, in which a carpool means a match between driver and passenger, so a driver with two passengers form 2 carpools, and a driver with 3 passengers form 3 carpools, etcetera. Figure 5-13 shows the distribution of the suggested carpools by the model, based on the percentage of the total ride that a person does not drive, thus rides with someone else. So, in 120 cases, persons leave their car (nearly) the whole ride. To explain this, we refer to the example at the beginning of this section. Employees a and b share a ride for 30 minutes. In case employee a leaves his car at the meeting point, this carpool is $\frac{30}{100} = 30\%$ of total ride duration of employee a, and falls in the category 20-30. When employee b leaves his car at the meeting point, he leaves his car for $\frac{30}{50} = 60\%$ of total ride duration, and the carpool falls in the 50-60 category. So, in case two persons share a ride, the calculation is only made for the person who does not drive. We conclude that the model generates 277 carpools that have sufficient length (>50% of total ride duration). However, 234 of the suggested carpools have a relative short duration (≤50% of total ride duration). It is questionable whether an employee is willing to leave his car for the last 5 minutes of his ride, when the total ride duration is 30 minutes ($\frac{5}{30} = 17\%$).

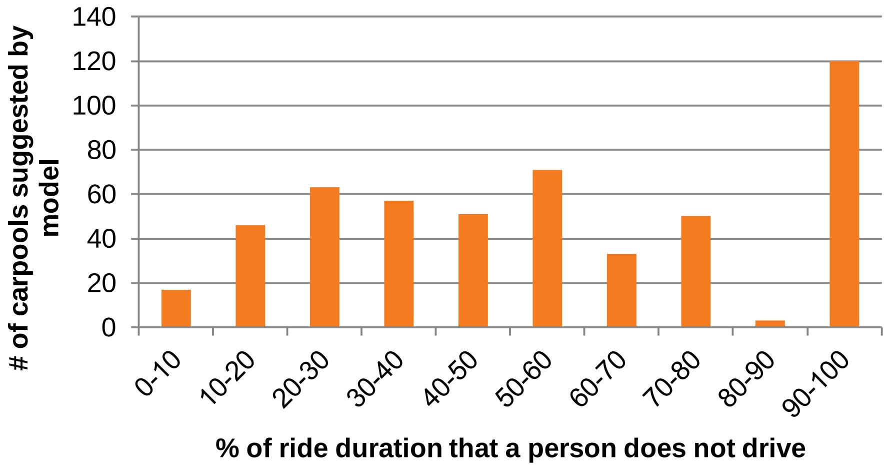


Figure 5-13: Distribution of the percentage of the time a person does not drive compared to total ride duration

Figure 5-14 plots the carpool potential of each day against the number of rides that day. We expect to see a positive relation between the number of rides and the carpool potential, because more rides should give the model more opportunities to find carpool matches. In this case, points should be scattered around a (imaginary) diagonal line or curve in the figure. However, when looking at Figure 5-14, we do not observe a clear relation between the number of rides and the carpool potential. So we conclude that, based on the data we collected, the number of rides per day is not necessarily an indicator of the carpool potential.

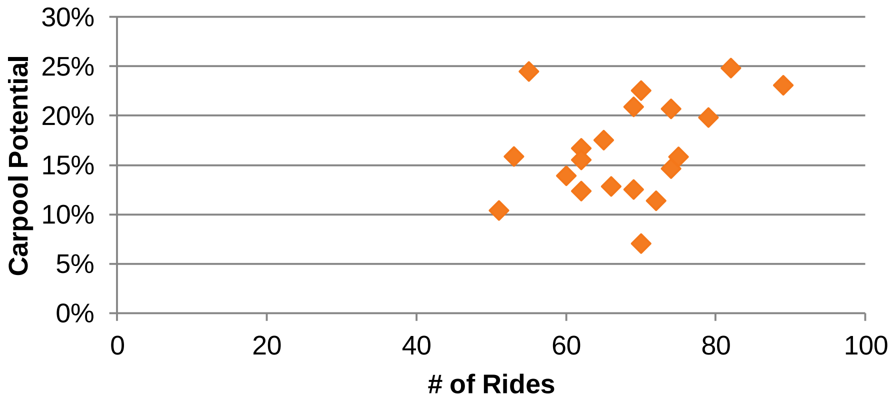


Figure 5-14: Carpool potential compared to the number of rides.

In Section 3.2, we discussed the work of Agatz et al. (2010), who argue that a ridematching problem becomes difficult to solve when riders can hop between drivers or drivers can pick up multiple riders. To indicate the computational complexity of our ridematching model, we show the computation times (together with the number of rides) for each day in Figure 5-15. The running time of the model varies between 1 and 7 minutes for a range of 50 to 90 rides. As discussed in Section 2.4, our matching algorithm should be an offline algorithm for which user do not expect immediate response. In this context, the running times of our ridematching model are acceptable.

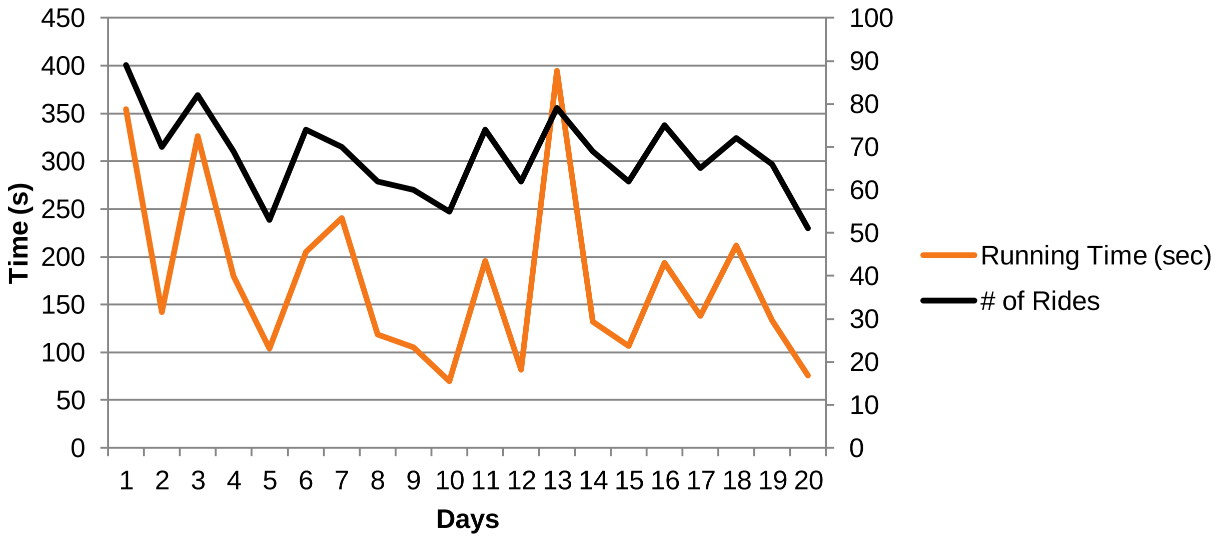


Figure 5-15: Running time of the model, and number of rides for each day.

Examining Figure 5-15, we observe that the trend lines of the number of rides and the computation times of the model have similar shapes. This indicates that the model's computation time depends on the number of rides. To check their relation, we plot the running time of each day against the number of rides that day in Figure 5-16. The points in the figure are clearly scattered around a (imaginary) diagonal line or curved line, which means there is a positive relation between the running time of the model and the number of rides. Based on the results, the number of rides can serve as an indicator for the computation time of the model.

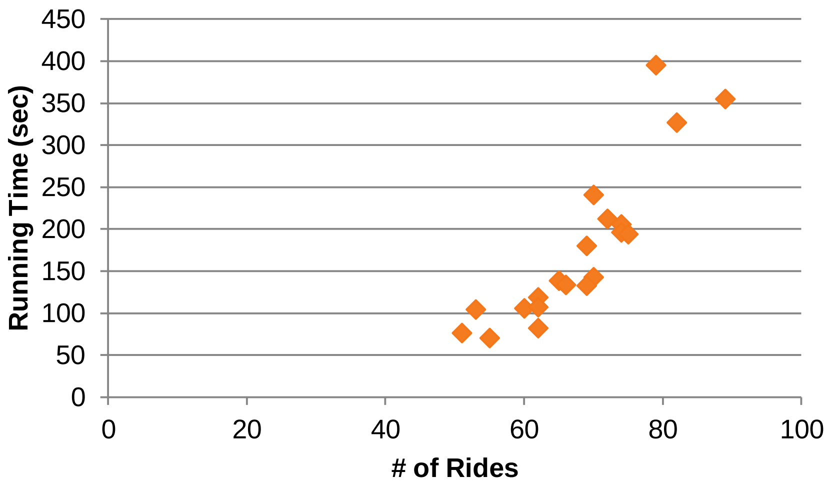


Figure 5-16: Running time of the model plotted against the number of rides.

In Section 5.1.2, we established a drawback of our shortest path computations that may lead to unrealistic carpool matches. Unrealistic carpool matches occur when the model suggests that person i rides with person j on edge (a, b), where node a is on one of the alternative routes of person j, and node b on the other alternative route of person j, while edge (a, b) does represent a real life shortest path edge of person i. Another unrealistic carpool match occurs when the model suggests that persons i and k ride with person j, and that person i needs to be picked up along one of the alternative routes of person j, while person k needs to be picked up along the other alternative route of person j. However, we can adjust the (parameters of the) ridematching model such that these situations cannot occur. When we make sure that the model only matches persons for a carpool when they have the same destination, we avoid the situation that edge (a, b) represent a real life shortest path edge of person i, while not being part of the real life shortest path of person j. Furthermore, when persons can maximally carpool with one other person during a ride, we exclude the possibility that person i needs to be picked up along one of the alternative routes of person j, while person k needs to be picked up along the other alternative route of person j. We do this by the following adjustments:

- We adjust the assumption that every city within a 10 minute reach of the travel time between origin and destination is on the shortest path, to the assumption that only cities within a 0 minute reach are on the shortest path. This means that, besides carpool parking lots, only the origin and destination cities are on the shortest path.
- Persons can maximally join one carpool per ride. This means that when persons carpool, they must have the same destination, because persons cannot hop to another carpool, and the only cities in the shortest path are origin and destination cities (see the first adjustment).
- Set maximum capacity of each person to 1. Persons cannot take more than 1 person simultaneously, which automatically means that persons cannot take more than 1 person per ride, because persons in a carpool always have the same destination (see the second adjustment).

Figure 5-17 shows the carpool potential for day 1 before and after the adjustments. It is important to realize that these adjustments (also) exclude feasible carpool possibilities. We make these adjustments to the model to show the effect on the carpool potential, when we want to be absolutely sure that the model does not suggest any unrealistic carpool possibilities. We conclude that the model is still able to generate significant savings (18%) in distance driven after the adjustments.

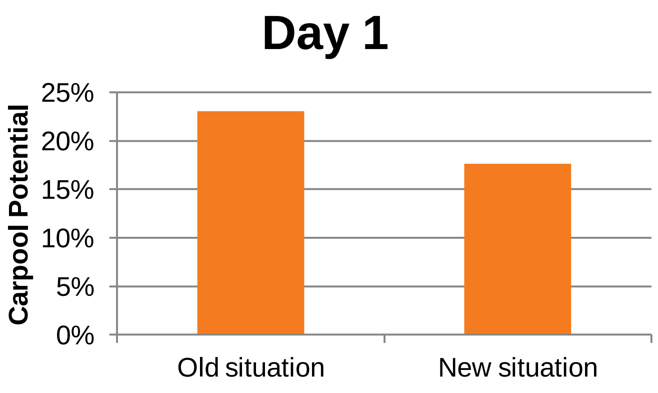


Figure 5-17: Carpool potential for day 1, before and after adjustments.

5.4 Comparison With Existing Apps

In Section 2.3, we discussed the functionalities of existing apps. We stated that the apps of ZGT and Flinc are not able to cope with variable meeting points and that *"to the best of our knowledge, no carpool apps exist that can recognize carpool possibilities where driver and passenger start the carpool at a certain meeting point chosen by the carpool app".* In this section, we compare our model with existing apps.

To illustrate the drawbacks of existing carpool apps, we use an example with a ride from Amsterdam to Enschede and a ride from Rotterdam to Enschede (see Section 2.3). The corresponding routes merge at Amersfoort (see Figure 2-6), but existing carpool apps do not recognize a carpool possibility. To make a comparison with the existing apps, we use this example to show the functionalities of our method. Table 5-5 displays the input we give to our method. Because our ridematching model includes return restrictions, we do not solely enter the rides to Enschede, but also the return rides. When we do not give the return rides, the model will not suggest a carpool, because person 1 or 2 is not able to return to his car, which he should leave near Amersfoort. Because our model also matches on overlap in time schedules, we enter equal (latest) arrival times at Enschede for both persons and comparable time schedules for the return ride. Shortest path computations, discussed in Section 5.1.2, determine the shortest paths between Amsterdam and Enschede, and Rotterdam and Enschede (also for the return rides). We insert these shortest paths, together with the person numbers, ride numbers, and latest arrival times, in our ridematching model.

Person	Ride	From	To	Latest Arrival Time
1	1	Amsterdam	Enschede	12:00
	2	Enschede	Amsterdam	18:00
2	1	Rotterdam	Enschede	12:00
	2	Enschede	Rotterdam	18:00

Table 5-5: Input to our method

In contrast to existing apps, our ridematching model does match persons 1 and 2 for a carpool. The model suggests that person 2 rides with person 1, and that they start the carpool at a carpool parking lot near Barneveld, see Figure 5-18. On the return ride person 2 rides with person 1 again until this meeting point.

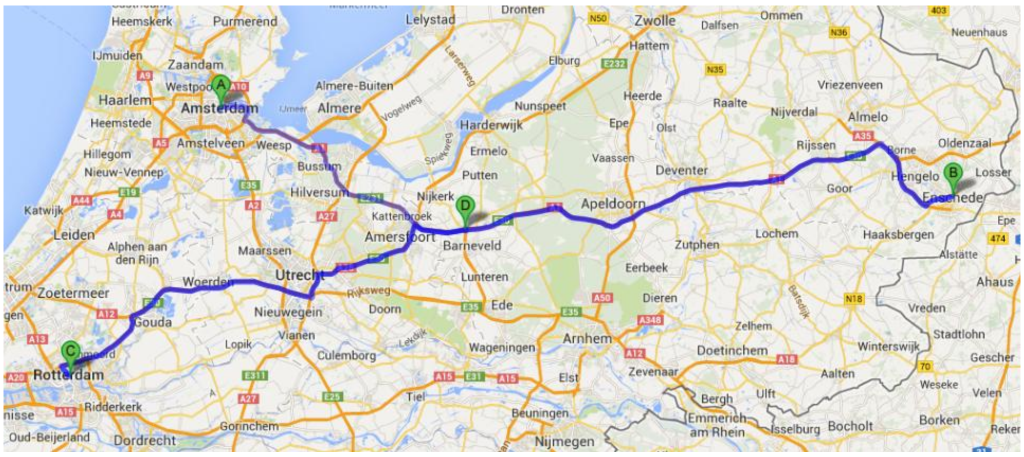


Figure 5-18: Our method suggest to meet at a carpool parking lot near Barneveld, when two rides merge at Amersfoort (Google).

In this section, we use an example with two rides that merge at a certain point (Amersfoort), while origins of both rides are not close to each other (Amsterdam and Rotterdam). In situations with comparable characteristics, our model is able to generate carpool matches that are not recognized by existing carpool apps. However, existing apps also generate carpool matches that are not recognized by our model. In Section 4.1, we mentioned that existing carpool apps enable the driver to take a detour in order to pick up a passenger, and that our model stick the driver and passenger to their routes. So in case two rides do not show overlap, but the origin of one ride lies within an acceptable reach of the other ride, existing apps will generate a carpool match, while our model does not. Figure 5-19 shows that the (shortest) routes from Rotterdam to Enschede and from Doetinchem to Enschede show (nearly) no overlap. These rides will not generate a match in our ridematching model. However, Doetinchem is located within an acceptable range of the alternative route from Rotterdam to Enschede (via Arnhem). Picking up a person in Doetinchem does not require a large detour for a person driving from Rotterdam to Enschede. Existing apps, such as Flinc, do generate a match between these two persons.

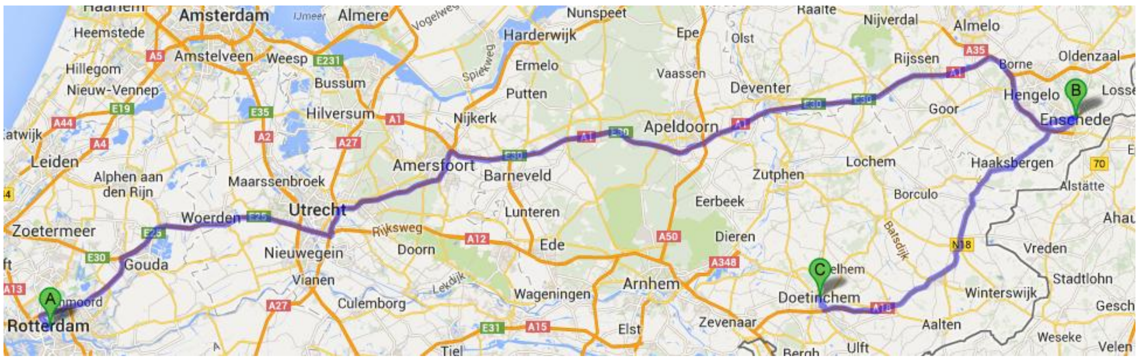


Figure 5-19: The routes from Rotterdam to Enschede and from Doetinchem to Enschede show no overlap (Google).

Other differences between our method and existing apps are:

- Existing apps make a strict distinction between driver and passenger. They assume that a passenger needs to be picked up at the address he provides. However, we assume every person (passenger and driver) has a car, thus can drive to a meeting point.
- Our model takes multiple rides per person into account, and existing apps only match one-way rides. As a consequence, our model only matches persons for a carpool when they are able to reach their destination(s), and are able to return to their cars (in case they left the car at a carpool parking lot) and eventually their homes. Existing apps only ensure that each person reaches the destination of the one-way ride.
- The ridematching model is designed as an offline tool, which runs once for all rides of a given day. Existing apps run their ridematching algorithm every time a ride is entered in the system and thus make use of an online algorithm. Due to this difference, we are able to determine an optimal solution (based on our model's objectives) for a given day, and existing apps encounter optimality problems when newly entered rides enable better carpool opportunities than suggested to the user. However, because we only determine one optimal solution, we only make one carpool suggestion. Existing apps provide multiple carpool suggestions, such that the user can choose the most suitable carpool for him.
- Some existing apps (such as the ZGT app) do not take time windows into account at all, when matching users for a carpool. Others (such as Flinc) suggest all carpool possibilities that fall into a time window set by the user. However, we assume each user wants to arrive as close as possible to his latest arrival time. We accept deviation from latest arrival times, until an earliest departure time. More importantly, we try to reduce waiting time during the ride (caused by hopping between carpools) to a minimum. We do not consider any deviation from latest arrival time at the destination as waiting time.

5.5 Summary

In this chapter, we elaborated on the characteristics of the inputs to the ridematching model, such as the shortest paths determined for the collected rides, we initialized the model's parameters, and we presented the results of our experiments. Furthermore, we compared the functionalities of our method with existing apps. In this section, we summarize the main aspects discussed in this chapter.

The rides collected amongst Significant personnel have two main characteristics:

- The rides are defined on city level. Rides occur between an origin city and a destination city.
- Time schedules of the rides are defined on an hourly level. Employees indicated the arrival time in a one-hour interval.

With Google Maps API and Bing Maps API, we calculate the travel times between all relevant carpool places (carpool parking lots and destinations of the ride collection). Based on these travel times and a person's origin city and destination city, we calculate the edges that make up the shortest path between these cities, which are connections between relevant carpool places. These shortest path computations are based on assumptions and may not fully represent a real life situation. However, the shortest paths serve as input to the ridematching model, therefore the drawbacks of the method to determine these shortest paths does not affect the validity of the ridematching model.

We initialize the weights of the model's objectives by running the model several times for one day in our data, and change the weights with each run (without changing any other settings). These experiments lead to a weight configuration that leads to optimal values for the model's main objectives: minimizing total waiting time and maximizing carpool time.

After initialization of the model's parameters, we run the model for each day of the data collection. Based on these experiments, we conclude that Significant employees could have saved 7% (day 2) to 25% (day 3) of the distance driven. On the 1416 rides we collected, 511 rides could have been (partly) saved by carpooling. We conclude that the model generates 277 carpools that have sufficient length (>50% of total ride duration). However, 234 of the suggested carpools have a relative short duration (≤50% of total ride duration). The carpools with a relative short duration are difficult to perform in real life, because employees will not leave their cars for relatively short carpools. When we compare the performance of our method with that of existing apps, we conclude that our method is able to recognize carpool possibilities that are not recognized by existing carpool apps and vice versa.

6. Conclusions & Recommendations

In this chapter, we provide the conclusions of our research in Section 6.1. Subsequently, Section 6.2 discusses the limitations of our method. Section 6.3 describes the recommendations for implementing our method in a real life carpool app. Finally, Section 6.4 suggests areas for further research.

6.1 Conclusions

In this section, we give the conclusions of our research. The reason for conducting this research was that Significant's employees lack insight into their colleagues' whereabouts and therefore only accidentally share a ride. Therefore, we aimed at developing a method that facilitates carpooling amongst employees, or any closed user group, by matching persons for shared rides. Input to the method, consisting of users' ID, starting points, destinations, and dates and times of the rides, should be derived from employees' agendas. The method should generate a concrete carpooling advice including participants, meeting point, possible drop-off point, and time schedules of the carpools.

The method should form the matching algorithm of a desired carpool app and be able to recognize carpool possibilities where driver and passenger meet at a meeting point chosen by the method. Further functionalities of the method should be to minimize detours, match travellers based on their time schedules, and take into account driver's capacity. We suggest an offline matching algorithm, which means that the method matches drivers and passengers once all requests are entered in the system. We conclude that no carpool app exists that is able to meet all described functionalities of the method.

In contrast to prevailing methods in literature or existing carpool apps, we suggest a method for carpooling without detours, which can be done by matching persons with actual overlapping routes. We divide this method in two components: shortest path computations and a ridematching model. In our research, we assume shortest path computations are known and we therefore focus on the development of the ridematching model. The main objective of this ridematching model is to minimize the distance driven by all users. Given the shortest path and time window of each user, the model matches persons with (partly) overlapping shortest paths and time windows for a carpool, under the condition that certain constraints, such as capacity and return restrictions, are met.

To examine the applicability of the method we suggest, we performed experiments on rides driven by Significant personnel. Based on these experiments, we conclude that Significant employees could have saved 7% (day 2) to 25% (day 3) of the distance driven. On the 1416 rides we collected, 511 rides could have been (partly) saved by carpooling. In contrast to our expectations, our conclusion is that days with a high amount of rides do not necessarily lead to a high carpool potential. We do observe a positive relationship between the number of rides and the calculation time of the model. For the daily number of rides of Significant personnel (50 to 90 rides), the running time varies between 1 and 7 minutes, which is an acceptable time for an offline matching algorithm. To exclude the possibility of unrealistic carpool matches, we adjust the model. However, these adjustments also lead to the exclusion of potentially feasible carpools. After these adjustments, we conclude carpool potential drops with 5% for day 1 of the data set to a level of 18%, which is still a significant saving on the rides driven by Significant personnel.

When we compare the functionalities of our method with that of existing apps, we conclude that there is a clear distinction in functionalities between our method and existing apps. In case two rides show overlap, while origins of both rides are not close to each other, our method is able to generate carpool matches that are not recognized by existing carpool apps. However, when two rides do not show overlap, but the origin of one ride lies within an acceptable reach of the other ride, existing apps will generate a carpool match, while our model does not.

The result of this research is a method that can be used in the development of a carpool app. Although the method is not yet extensively tested, we believe it can make a valuable contribution to the reduction of the distance driven by the employees of Significant, or any closed user group with the characteristic that every person has a car and is able to drive to a certain carpool meeting point. The method minimizes discomfort, such as waiting times, and excludes detours for each user.

6.2 Limitations

In this section, we discuss the limitations of our method. As discussed in Section 4.2, our method consists of two components: shortest path computations and the ridematching model, see Figure 6-1. The shortest paths serve as input for the ridematching model. In this research we focus on the development of the ridematching model. However, in our experiments we also performed shortest path computations to generate input for the ridematching model. Section 6.2.1 discusses the limitations of the shortest path computations we performed in our experiments. Section 6.2.2 elaborates on the limitations of the ridematching model.

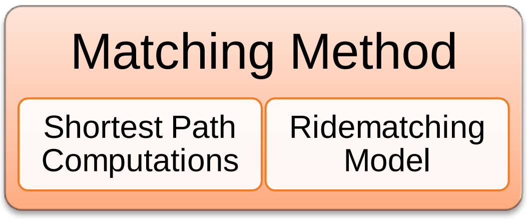


Figure 6-1: Our matching method consists of shortest path computations and our ridematching model

6.2.1 Limitations Shortest Path Computations

To conduct experiments on the ridematching model, we had to determine the edges that make up the shortest path between two cities. In the last part of Section 5.1.2, we discussed drawbacks of the shortest path computations we perform for the collected rides. The major drawback of our computations is that they are based on assumptions. This may lead to misleading representations of real life situations. In our experiments, travel times between origin and destination are larger than in real life, and edges of the shortest path may not fully represent edges driven in a realistic situation. However, because we apply the same shortest path computations to every ride, we provide consistent input to the ridematching model, enabling the model to generate representative carpool matches.

Another drawback of the assumptions we make when computing shortest paths is that when a ride from a to b has two (or more) alternative routes with comparable travel times, our method will appoint nodes along both routes to the shortest path from a to b. This can cause the model to suggest unrealistic carpools. In Section 5.3, we made adjustments to the model such that we exclude the possibility of infeasible carpool suggestions.

6.2.2 Limitations Ridematching Model

Section 5.4 discussed the differences between our model and existing carpool apps. These differences illustrate limitations of our model. The first limitation is that the ridematching model only matches shortest paths, and does not allow detours for the users. However, due to the characteristics of the collected rides (on a city level) and shortest path computations (all places within a 5 or 10 minute range), our experiments do allow a small detour. For example, when two persons travel to the same city, the model can recognize a match, because our shortest path computations lead these persons to the same point. In practice, these persons do not necessarily need to be at the same building, so the driver has to make a detour within the destination city to drop off the passenger.

A second limitation of the ridematching model is that it is designed as an offline tool, which runs once for all rides of a given day. The model determines an optimal solution (based on our model's objectives) for a given day, but this takes time. In Section 5.3, we compared the number of rides for each day with the running time of the model each day. We conclude that the running times of our ridematching model are acceptable for the range of rides we collected for each day. However, when we apply the ridematching model to a much larger set of rides, the running times will increase. Besides this, because we only determine one optimal solution, we only make one carpool suggestion. With our ridematching model, users do not have a choice.

The third limitation is that the model assumes every person has a car, thus can drive to a meeting point. However, when the user drives to a meeting point to join a carpool, he has to return to this point at a later ride. This means that our model only matches persons in case of one-way rides, when the passenger is picked up at his home address. Unfortunately, a user cannot enter that he must be picked up at his home address. Furthermore, the model is not usable for users without a car, because the model does not take into account any other forms of transport (such as public transport) besides the car.

Concerning time, we assume that persons would like to arrive as close as possible to their latest arrival times. However, in real life also situations occur when persons would like to arrive as close as possible to earliest arrival times, for example when an employee has a ride from the office to his home. Furthermore, we do not consider deviation from latest arrival time at destination as waiting time, but in reality arriving earlier than latest arrival time can lead to waiting, for example in case of arriving 30 minutes earlier for an appointment at the customer.

In Section 5.3, we stated that 234 of the 511 suggested carpools, have a relatively short duration, which is a limitation of the model. The model is not able to filter out the relatively short carpools, and only suggest carpools that have a sufficient length.

6.3 Recommendations

This section describes the recommendations for implementing our method in a real life carpool app.

An important assumption we make in conducting this research is that travellers are willing to carpool. We assume no further barriers to carpooling exist, except an unclear view on the carpool possibilities amongst acquaintances. However, from conversations with employees of Significant we learn that there are other barriers. Some employees indicated that they would rather not participate in carpools, because they are very fond of their privacy during their ride from home to work, or vice versa. Furthermore, they fear to form a carpool with colleagues that do not have much in common. Consequently, we recommend to pay attention to human resistance, when introducing the use of a carpool app at Significant, or any other closed user group with comparable characteristics. Persons can be motivated by financial or nonfinancial incentives. For instance, persons that form a carpool can be awarded with a share of the costs (gasoline, etcetera) they save, or with points that can be redeemed for products or services (for example extra vacation time or clothing).

Furthermore, we recommend to involve potential users in the implementation process of a carpool app. In the experiments we initialize parameters, such as maximum waiting time and maximum capacity. However, it is important to determine the settings of the model according to the wishes of the potential user. In Section 6.2, we state that the model is not able to filter out the relatively short carpools. We also recommend to discuss with potential users as of which percentage of their total ride they find it acceptable to join a carpool.

The method we suggest in this research is designed as an offline matching algorithm. In Section 2.4, we suggest to run the algorithm once a day when all rides for the next day are presumably known, for example at 6 pm. During a run, the model determines an optimal solution (based on our model’s objectives) for a given day. As a consequence, we only make one carpool suggestion, and users do not have a choice. When we aim at developing a carpool app that gives each user immediate response after entering the required data (such as starting point, destination, and time window), and allows the user to choose between different carpool offers, we recommend to use a heuristic approach that requires less computation time than our model.

We determine the travel times between all relevant nodes by making use of Google Maps API and Bing Maps API. A drawback of these APIs is that they do not allow an unlimited amount of requests for the travel time between two given places. So when we want to implement the carpool method in a real carpool app, we recommend to start contacting companies (Google, Microsoft, TomTom) that have access to traffic data, such as travel time between two places, because reliable travel times are crucial for the method. Furthermore, we recommend to critically consider which carpool parking lots are suitable for switching between cars. For this research we only relied on a list of Carpoolplein.nl with carpool parking lots in the Netherlands.

6.4 Further Research

To the best of our knowledge, no carpool apps or methods in literature exist that approach the ridematching problem in the way our method does. Because of this, our method requires further research in several ways. In this section, we discuss the issues for further research.

In this research we suggest a carpool method, consisting of shortest path computations and a ridematching model. This model is designed according to the functionalities desirable for a carpool app at Significant, as described in Chapter 4. The model is validated with Significant related rides collected in a four-week period. We manually checked whether the ridematching model suggested the carpools that it should, according to the functionalities. As far as we know, the model functions according to this functionalities. Further research should assess the model’s performances on other ride sets (more rides per day, other origin/destination cities) than the ones we used.

In Section 5.1.2, we described the shortest path computations. These computations are based on assumptions, which sometimes lead to unrealistic shortest paths. We make adjustments to the model, such that infeasible carpool suggestions due to drawbacks in shortest path computations, do not occur. However, we hereby also exclude feasible carpool possibilities. When shortest path computations are not based on assumptions, but on facts, they do not lead to unrealistic shortest paths, and we do not need to make adjustments to the model. So, we recommend to do further research in shortest path determination.

We did not take into account the possibility to travel by public transport in conducting this research. However, the adoption of public transport possibilities in the model will further reduce the distance driven by car. Furthermore, it will increase the number of potential meeting points with, for example, train stations. Following on this, we assumed every user of our carpool method has a car and is able to drive to a given meeting point. Adopting the option to travel by public transport, means that carless users are also able to travel to a certain meeting point (by train or bus). We recommend further research to extend the method with public transport possibilities and functionalities for carless users.

A difference between existing apps and the method that we suggest, is that existing apps allow the driver to make a detour to pick up a passenger, and our model stick the driver and passenger to their routes. From conversations with the employees of Significant, we know that persons do not always drive the same route from, for example, Barneveld to Utrecht. There may be three different routes that all have an acceptable travel time. Further research should indicate whether our method is able to take into account all of these three different routes in matching persons for a carpool.

The ridematching model matches persons when their time schedules can be adjusted such that they have minimum waiting time at the meeting point of the carpool (and minimum deviation from latest arrival times). However, we did not take into account any time buffers for any variable delay incurred on the road, such as traffic congestion. Besides this, we determine a matrix with offline travel times between all relevant nodes, in which 'offline' means: travel time under normal conditions. When the method is used in real life situations, we recommend to extend the model with time buffers, or to use online travel times. For both options further research is required.

In our literature research, we did not encounter any papers that introduce methods for matching overlapping paths. We designed a ridematching model that is able to match users that travel on the same edges, given the users' shortest paths consisting of subsequent edges. We build the model in three stages, where each stage extends the previous stage with further carpool characteristics, such as matching on time schedules, and returning to parked cars. However, we believe the ridematching model could form a basis for other problems where matching of paths is relevant, when further research is done. For example, the ridematching model with time and capacity constraints could be applicable to merging freight transport.

References

01 Abraham, I., D. Delling, A. Fiat, A. V. Goldberg and R. F. Werneck (2013). "Highway Dimension and Provably Efficient Shortest Path Algorithms." Silicon Valley.

02 Agatz, N., A. Erera, M. Savelsbergh and X. Wang (2010). Sustainable passenger transportation: Dynamic ride-sharing, Erasmus Research Institute of Management (ERIM).

03 Akiba, T., Y. Iwata and Y. Yoshida (2013). "Fast Exact Shortest-Path Distance Queries on Large Networks by Pruned Landmark Labeling." arXiv preprint arXiv:1304.4661.

04 Amey, A., J. Attanucci and R. Mishalani (2011). "Real-Time Ride-sharing." Transportation Research Record: Journal of the Transportation Research Board **2217**(1): 103-110.

05 Barbucha, D. (2013). "Agent-Based Simulator for Travellers Multimodal Mobility." Advanced Methods and Technologies for Agent and Multi-Agent Systems **252**: 81.

06 Bast, H., D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner and R. Werneck (2014). "Route Planning in Transportation Networks."

07 Bast, H., S. Funke, D. Matijevic, P. Sanders and D. Schultes (2007). In Transit to Constant Time Shortest-Path Queries in Road Networks. ALENEX.

08 Bauer, R., D. Delling, P. Sanders, D. Schieferdecker, D. Schultes and D. Wagner (2010). "Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm." Journal of Experimental Algorithmics (JEA) **15**: 2.3.

09 Bazaraa, M. S., J. J. Jarvis and H. D. Sherali (2011). Linear programming and network flows, Wiley. com.

10 Beker, I., V. Jevtic and D. Dobrilovic (2011). Using shortest-path algorithms for forklift route planning and optimization. Proceedings of the XV International Scientific Conference on Industrial Systems (IS'11), Buy this book from publisher.

11 Bellemans, T., S. Bothe, S. Cho, F. Giannotti, D. Janssens, L. Knapen, C. Körner, M. May, M. Nanni and D. Pedreschi (2012). "An Agent-Based Model to Evaluate Carpooling at Large Manufacturing Plants." Procedia Computer Science **10**: 1221-1227.

12 Bellman, R. (1956). On a routing problem, DTIC Document.

13 Bit-Monnot, A., M.-J. Huguet, M.-O. Killijian and C. Artigues (2013). "Carpooling: the 2 Synchronization Points Shortest Paths Problem."

14 Bonsall, P. (1981). "Car sharing in the United Kingdom: a policy appraisal." Journal of Transport Economics and Policy: 35-44.

15 Brownstone, D. and T. F. Golob (1992). "The effectiveness of ride-sharing incentives: Discrete-choice models of commuting in Southern California " Regional Science and Urban Economics **22**(1): 5-24.

16 Chan, N. D. and S. A. Shaheen (2012). "Ride-sharing in north america: Past, present, and future." Transport Reviews **32**(1): 93-112.

17 Chaube, V., A. L. Kavanaugh and M. A. Perez-Quinones (2010). Leveraging social networks to embed trust in rideshare programs. System Sciences (HICSS), 2010 43rd Hawaii International Conference on, IEEE.

18 Coltin, B. J. and M. Veloso (2013). Towards ride-sharing with passenger transfers. Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems.

19 Correia, G. and J. M. Viegas (2011). "Carpooling and carpool clubs: Clarifying concepts and assessing value enhancement possibilities through a Stated Preference web survey in Lisbon, Portugal." Transportation Research Part A **45**(2): 81-90.

20 Deakin, E., K. T. Frick and K. M. Shively (2010). "Markets for Dynamic Ride-sharing?" Transportation Research Record: Journal of the Transportation Research Board **2187**(1): 131-137.

21 Delling, D., P. Sanders, D. Schultes and D. Wagner (2009). Engineering route planning algorithms. Algorithmics of large and complex networks, Springer**:** 117-139.

22 Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs." Numerische mathematik **1**(1): 269-271.

23 Dill, J. and E. Wardell (2007). "Factors Affecting Worksite Mode Choice: Findings from Portland, Oregon." Transportation Research Record: Journal of the Transportation Research Board **1994**(1): 51-57.

24 Drews, F. and D. Luxen (2013). Multi-Hop Ride Sharing. Sixth Annual Symposium on Combinatorial Search.

25 Elizabeth, S. and L. Sujatha (2012). "Fuzzy shortest path problem based on level λ-triangular LR fuzzy numbers." Advances in Fuzzy Systems **2012**: 1.

26 Evans, J. R., W. K. Jackson, G. J. Westerbeck and M. P. Thomas (1985). "Planning and analysis of a ride-sharing evaluation study " Socio-Economic Planning Sciences **19**(1): 41-49.

27 Ferguson, E. (1997). "The rise and fall of the American carpool: 1970–1990." Transportation **24**(4): 349-376.

28 Flinc. "flinc brochure." Retrieved December 20, 2013, from https://flinc.org/corporate/brochure.

29 Ford, D. and D. R. Fulkerson (2010). Flows in networks, Princeton university press.

30 Geisberger, R., D. Luxen, S. Neubauer, P. Sanders and L. Volker (2009). "Fast detour computation for ride sharing." arXiv preprint arXiv:0907.5269.

31 Geisberger, R., P. Sanders, D. Schultes and D. Delling (2008). Contraction hierarchies: Faster and simpler hierarchical routing in road networks. Experimental Algorithms, Springer**:** 319-333.

32 Ghoseiri, K., A. E. Haghani, M. Hamedi and M.-A. U. T. Center (2011). Real-time rideshare matching problem, Mid-Atlantic Universities Transportation Center.

33 Giuliano, G., D. W. Levine and R. F. Teal (1990). "Impact of high occupancy vehicle lanes on carpooling behavior." Transportation **17**(2): 159-177.

34 Goldberg, A. V. and C. Harrelson (2005). Computing the shortest path: A search meets graph theory. Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics.

35 Google. "Google Maps." Retrieved December 23, 2013, from https://maps.google.com/.

36 Gruebele, P. (2008). "Interactive system for real time dynamic multi-hop carpooling." Global Transport Knowledge Partnership.

37 Gunelius, S. (2011). What Is an API and Why Does It Matter?

38 Handke, V. and H. Jonuschat (2013). Carpooling. Flexible Ride-sharing, Springer**:** 13-40.

39 Hart, P. E., N. J. Nilsson and B. Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths." Systems Science and Cybernetics, IEEE Transactions on **4**(2): 100-107.

40 Herbawi, W. and M. Weber (2011). Evolutionary multiobjective route planning in dynamic multi-hop ride-sharing. Evolutionary Computation in Combinatorial Optimization, Springer**:** 84-95.

41 Herbawi, W. and M. Weber (2012). The ridematching problem with time windows in dynamic ride-sharing: A model and a genetic algorithm. Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE.

42 Jacobson, S. H. and D. M. King (2009). "Fuel saving and ride-sharing in the US: Motivations, limitations, and opportunities." Transportation Research Part D: Transport and Environment **14**(1): 14-21.

43 Karp, R. M. (1992). On-line algorithms versus off-line algorithms: How much is it worth to know the future? IFIP Congress (1).

44 Karp, R. M., U. V. Vazirani and V. V. Vazirani (1990). An optimal algorithm for on-line bipartite matching. Proceedings of the twenty-second annual ACM symposium on Theory of computing, ACM.

45 Karzoo. Website of Karzoo. Retrieved November 22, 2013, from http://www.karzoo.nl/en/companies-solutions/carpool-solutions.

46 Kleiner, A., B. Nebel and V. A. Ziparo (2011). A mechanism for dynamic ride sharing based on parallel auctions. Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One, AAAI Press.

47 Knopp, S., P. Sanders, D. Schultes, F. Schulz and D. Wagner (2007). Computing Many-to-Many Shortest Paths Using Highway Hierarchies. ALENEX.

48 Massaro, D. W., B. Chaney, S. Bigler, J. Lancaster, S. Iyer, M. Gawade, M. Eccleston, E. Gurrola and A. Lopez (2009). Carpoolnow-Just-in-Time Carpooling without Elaborate Preplanning. WEBIST.

49 Minett, P. and J. Pearce (2008). Flexible Carpooling: Challenging the Ride Matching Paradigm. Saving Energy by Making it Easier to Share Rides, Auckland, NZ: Trip Convergence Ltd.

50 Minett, P. and J. Pearce (2011). "Estimating the energy consumption impact of casual carpooling." Energies **4**(1): 126-139.

51 Morency, C. (2007). "The ambivalence of ride-sharing." Transportation **34**(2): 239-253.

52 Oost, V. "Carpoolplein.nl." from http://www.carpoolplein.nl/autonavigatie.html.

53 Pijls, W. and H. Post (2009). "A new bidirectional search algorithm with shortened postprocessing." European Journal of Operational Research **198**(2): 363-369.

54 Pohl, I. (1971). "Bi-Directional Search." Machine Intelligence **6**.

55 Pyrga, E., F. Schulz, D. Wagner and C. D. Zaroliagis (2004). Experimental Comparison of Shortest Path Approaches for Timetable Information. ALENEX/ANALC, Citeseer.

56 Rosenbloom, S. (1978). "Peak-period traffic congestion: A state-of-the-art analysis and evaluation of effective solutions." Transportation **7**(2): 167-191.

57 Sanders, P. and D. Schultes (2005). Highway hierarchies hasten exact shortest path queries. Algorithms–Esa 2005, Springer**:** 568-579.

58 Skiena, S. S. (2008). Graph Problems: Polynomial-Time. The Algorithm Design Manual, Springer**:** 475-522.

59 Vanoutrive, T., E. Van De Vijver, L. Van Malderen, B. Jourquin, I. Thomas, A. Verhetsel and F. Witlox (2012). "What determines carpooling to workplaces in Belgium: location, organisation, or promotion?" Journal of Transport Geography **22**: 77-86.

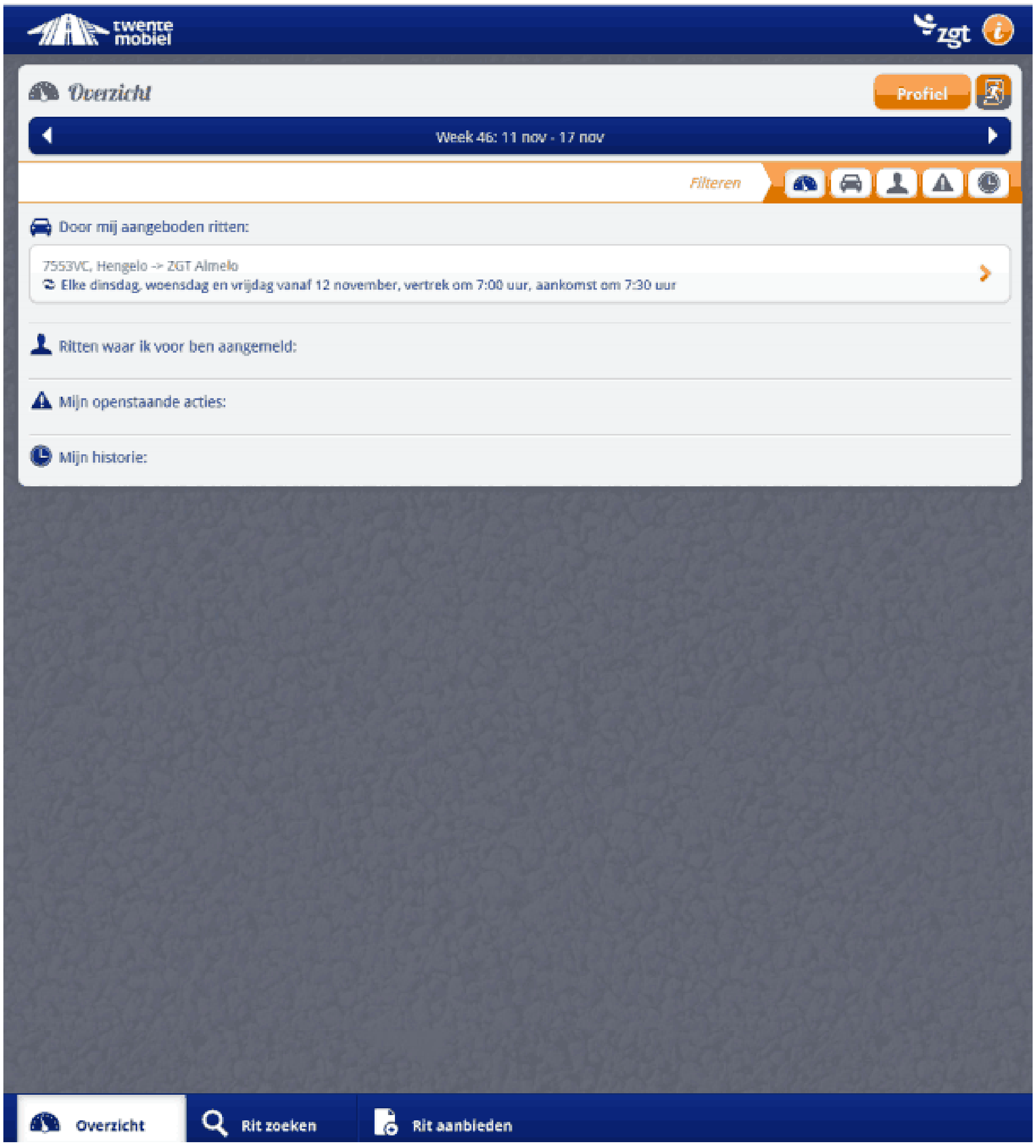
60 Wagner, D. and T. Willhalm (2003). Geometric speed-up techniques for finding shortest paths in large sparse graphs, Springer.

61 Winston, W. L. (1994). Solutions Manual: Operations Research: Applications and Algorithms, Third Edition : Introduction to Mathematical Programming : Applications and Algorithms, Second Edition, Duxbury Press.

Appendices

Appendix A

Screenshots carpool app ZGT



twente mobiel

zgt

Overzicht

Profiel

Week 46: 11 nov - 17 nov

Filteren

Door mij aangeboden ritten:

7553VC, Hengelo -> ZGT Almelo
Elke dinsdag, woensdag en vrijdag vanaf 12 november, vertrek om 7:00 uur, aankomst om 7:30 uur

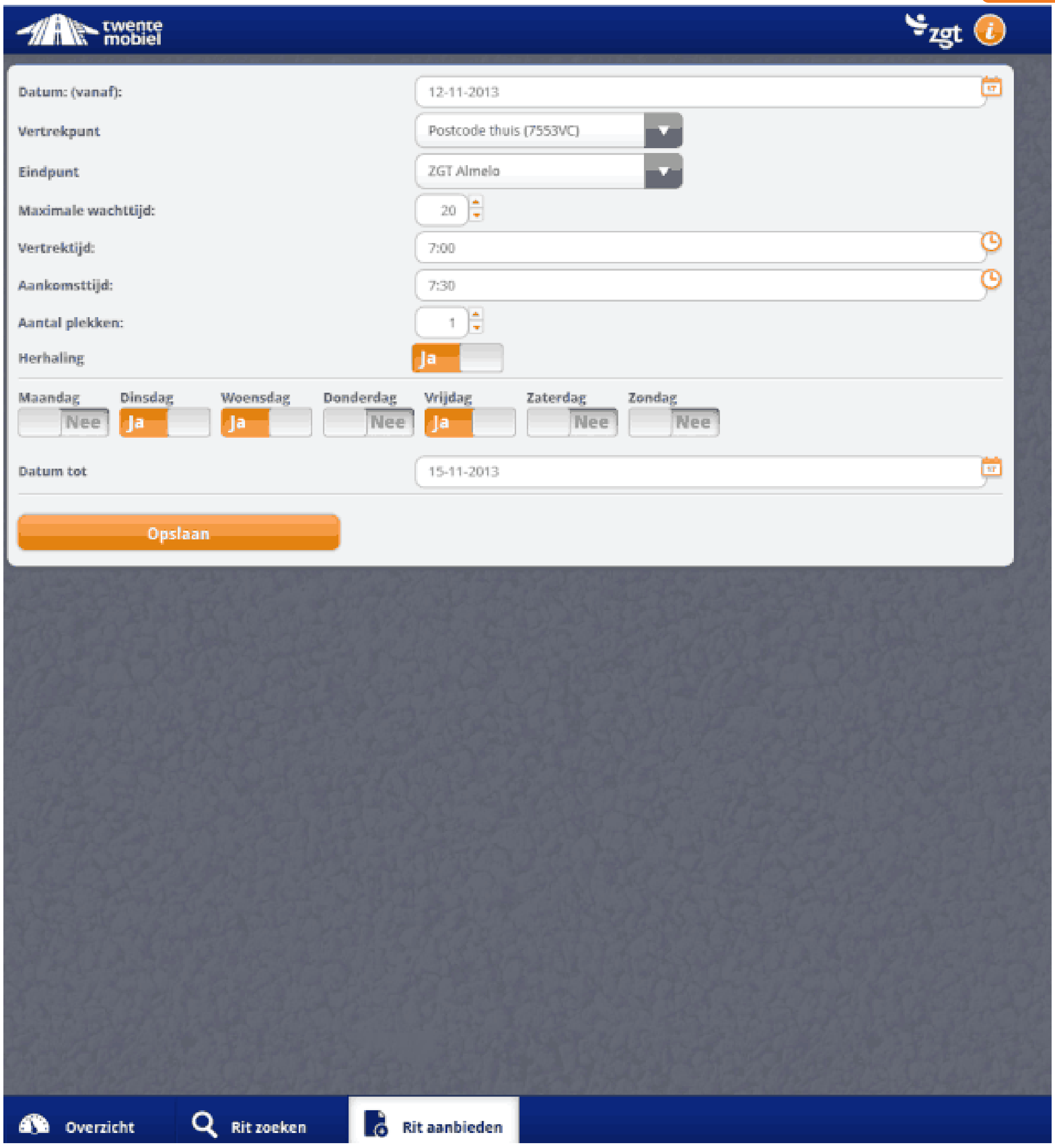
Ritten waar ik voor ben aangemeld:

Mijn openstaande acties:

Mijn historie:

Overzicht | Rit zoeken | Rit aanbieden

The homepage gives an overview of the rides we offered, the rides for which we subscribed (to drive along), passenger requests that we still have to accept/reject and carpooling history.



Datum: (vanaf):	12-11-2013
Vertrekpunt	Postcode thuis (7553VC)
Eindpunt	ZGT Almelo
Maximale wachttijd:	20
Vertrektijd:	7:00
Aankomsttijd:	7:30
Aantal plekken:	1
Herhaling	Ja

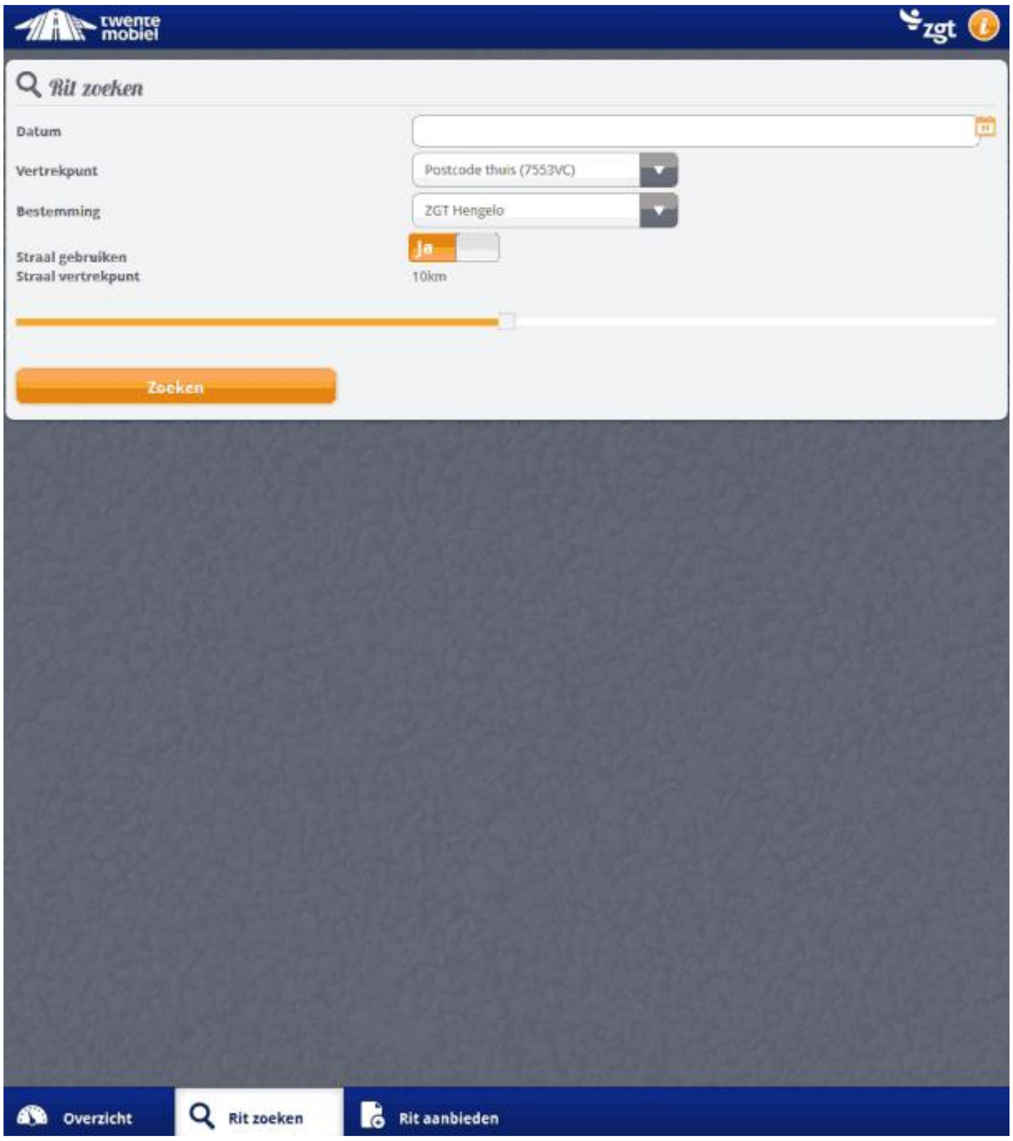
Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag	Zondag
Nee	Ja	Ja	Nee	Ja	Nee	Nee

Datum tot	15-11-2013

At the ride offer page, we can submit a ride we will take in the near future. After submitting, colleagues can find our ride and request to share this ride. In top-bottom order (see the figure above), we have to fill in the following lines:

- The date of our ride
- The starting point of the ride
- The destination of the ride
- Maximum waiting time. With this maximum waiting time we determine the flexibility of our ride. When we want to leave at, for example, 07.00 am and set our maximum waiting time at 15 minutes, we indicate that our latest departure time is 07.15 am. So people interested in joining our ride, have to accept a departure between 07.00 and 07.15 am. This maximum waiting time is only taken into account when a ride is offered from a ZGT location, because then both driver and passenger will be at the same location and thus are able to wait for each other.
- Time of departure from our starting point
- Time of arrival at our submitted destination

- Available spots in the car. When we only want to carpool with one colleague, we fill in 1.
- Whether the ride is recurring or not. When the ride we offer, is a standard ride that we take every week or multiple times a week, we can indicate that the ride is recurring. After this, we have to fill in which days this ride is taken and the end date of the series of rides.



Rit zoeken

Datum	
Vertrekpunt	Postcode thuis (7553VC)
Bestemming	ZGT Hengelo
Straal gebruiken	Ja
Straal vertrekpunt	10km

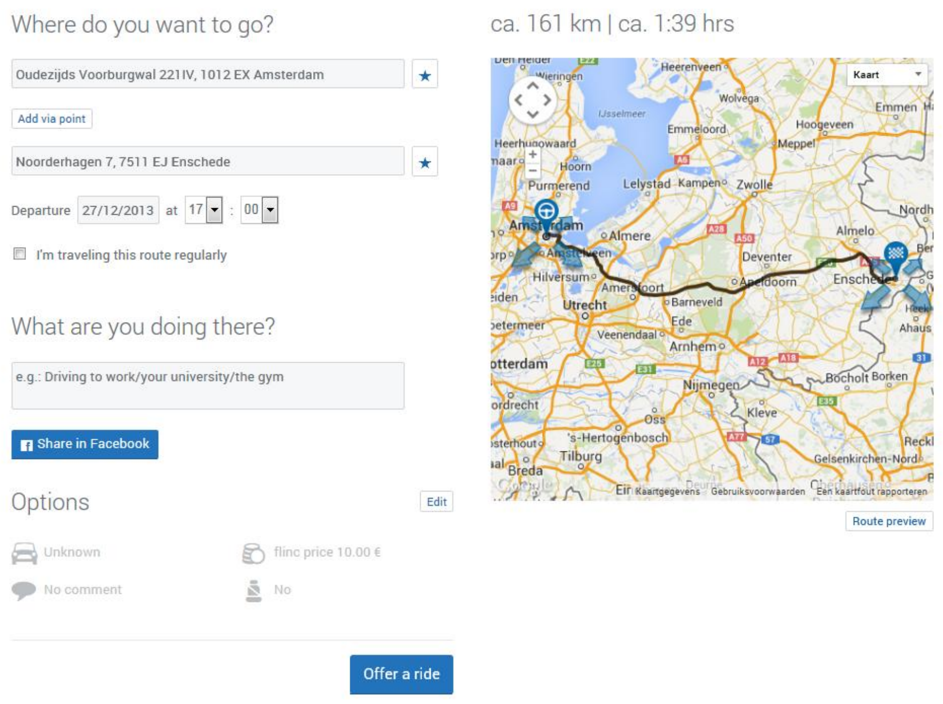
At the ride search page, we can set some constraints for the system in searching for offered rides. In top-bottom order (see figure above), we can fill in the following lines:

- The date when we need a ride
- Our starting point
- Our destination
- Whether we want to apply a maximum radius to our ride search. In case we apply a maximum radius, the app will draw a circle around our starting point or destination with the radius we set. All rides departing from or arriving at a location outside this circle will not be shown in the search results. For example, we seek a ride from home to ZGT Hengelo and fill in a maximum radius of 10 kilometres. Then the app will only show the offered rides to ZGT Hengelo with starting points within 10 kilometres of our house (at the day we submitted).

Appendix B

Ride offer and ride search with Flinc carpool app

For testing the app of Flinc, we use a base ride from Amsterdam to Enschede, that departs December 27, 2013 at 5 pm. With our Flinc account, we offer this ride to the ride-sharing network (see the figure below).


Where do you want to go?

Oudezijds Voorburgwal 221 IV, 1012 EX Amsterdam

Add via point

Noorderhagen 7, 7511 EJ Enschede

Departure 27/12/2013 at 17 : 00

I'm traveling this route regularly

What are you doing there?

e.g.: Driving to work/your university/the gym

Share in Facebook

Options Edit

Unknown | flinc price 10.00 €

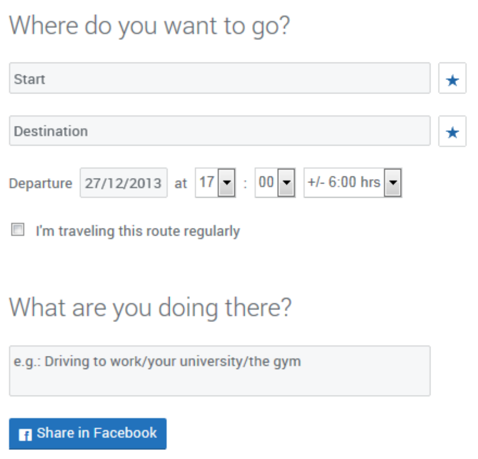
No comment | No

Offer a ride

ca. 161 km | ca. 1:39 hrs

Route preview

As the figure above shows, we have to fill in the starting point, destination, departure date and time for offering the ride. So the process of offering a ride is comparable to that of the ZGT app (Appendix A), because both tools require a starting point, destination, date and time of the ride. The figure below shows part of the ride searching page of Flinc.


Where do you want to go?

Start

Destination

Departure 27/12/2013 at 17 : 00 +/- 6:00 hrs

I'm traveling this route regularly

What are you doing there?

e.g.: Driving to work/your university/the gym

Share in Facebook

The ride search page is similar to the ride offer page, except the extra indicator at 'departure'. The ride seeker has to indicate the tolerance of his departure time. In the figure above, the ride seeker is willing to leave 6 hours earlier or later than his initial departure time.

Time criterion

As we describe in the previous section, we offer a ride from Amsterdam to Enschede. For determining the criteria Flinc uses to match offered rides with ride seekers, we then search for this ride with varying input parameters (date, time, starting point and destination). First, we search for rides from Amsterdam to Enschede with varying date and time (see the table below).

Searched ride	Match with offered ride Amsterdam-Enschede, December 27 at 5 pm?
Amsterdam-Enschede, December 27, 5 pm +/- 6 hrs	Yes
Amsterdam-Enschede, December 27, **2 pm +/- 3 hrs**	No
Amsterdam-Enschede, December 27, **2 pm +/- 3:15 hrs**	Yes
Amsterdam-Enschede, December 27, **4 pm +/- 1 hrs**	No
Amsterdam- Enschede, December 27**, 4 pm +/- 1:15 hrs**	Yes
Amsterdam- Enschede, December 27**, 9 am +/- 2 hrs**	No
Amsterdam-Enschede, **December 28**, 5 pm +/- 6 hrs	No

From the table above we can conclude that Flinc's matching criteria include a match in time windows. When an offered ride departs at a time outside the time window (9 am +/- 2 hrs) or at the edge of a time window of a searched ride (2 pm +/- 3 hrs), the Flinc app does not suggest a match.

Noteworthy is the situation in case of a match in time, but a mismatch in date. When this case occurs, the app recognizes a match. For example, when we offer a ride on December 27 at 11 pm and search for a ride on December 28 at 1 am with a tolerance of +/- 3 hrs, Flinc correctly determines a carpool possibility. So we can conclude that the first criterion of Flinc's matching algorithm is a time criterion that matches drivers and passengers based on overlap in time windows, regardless the date of the offered/searched rides. This overlap in time windows is defined as: the driver can arrive at the passenger's location within the time windows specified by the passenger.

Detour criterion

In the previous section, we searched for rides with non-varying starting point and destination (Amsterdam-Enschede), similar to the starting point ans destination of the offered ride. In this section, we vary the starting point and destination of the searched ride, in order to determine whether the Flinc app makes use of a starting point/ destination/ detour criterion. We search for the same offered ride as in the previous section. See the table below for the searched rides.

Searched ride	Detour	Match with offered ride Amsterdam-Enschede, December 27 at 5 pm?
Utrecht-Enschede, December 27, 5 pm +/- 6 hrs	19 km	Yes
Utrecht-Enschede, December 27, 5 pm +/- **0:30 hrs**	19 km	Yes
Utrecht-Enschede, December 27, 5 pm +/- **0:15 hrs**	19 km	No
Utrecht-Deventer, December 27, 5 pm +/- 6 hrs	30 km	Yes
Utrecht-Arnhem, December 27, 5 pm +/- 6 hrs	36 km	Yes
Nieuwegein-Deventer, December 27, 5 pm +/- 6 hrs	40 km	Yes
Nieuwegein-Almelo, December 27, 5 pm +/- 6 hrs	48 km (with help of Google maps)	No
Bunnik-Deventer, December 27, 5pm +/- 6 hrs	44 km (with help of Google maps)	No
Rotterdam-Enschede, December 27, 5 pm +/- 6 hrs	109 km (with help of Google maps)	No

The results of the searched rides in the table above give an indication of the criteria used by Flinc concerning starting point, destination and detours. Because the searched rides Utrecht-Deventer, Utrecht-Arnhem and Nieuwegein-Deventer generate a match with the offered ride Amsterdam-Enschede, we can conclude that the Flinc app does not encompass criteria regarding starting point and destination. With a high probability, we can state that the Flinc app uses a detour criterion instead of starting point/ destination criteria. Searched rides that require a detour larger than 40 kilometres (Nieuwegein-Almelo, Bunnik-Deventer and Rotterdam-Enschede) do not lead to a match. So the criterion is: searched rides that require a detour of about 40 kilometres or less are matched with an offered ride (when also meeting date and time criteria).

Noteworthy is the mismatch of the searched ride Utrecht-Enschede on December 27 at 5 pm with a tolerance of +/- 0:15 hrs. The the ride seeker is not able to depart from Utrecht between 4:45 pm and 5:15 pm, because the driver leaves Amsterdam at 5 pm and arrives in Utrecht at 5:28 pm. Summarizing, we can conclude the following about the matching process of the Flinc carpool app:

- The time constraint checks whether the arrival time of an offered ride at the starting point of a searched ride lies within the time windows of the searched ride.
- The detour constraint checks whether the searched ride requires a detour less than 40 kilometres for an offered ride.

Appendix C

Dijkstra's Algorithm

The method used in this appendix is derived from Winston (1994).

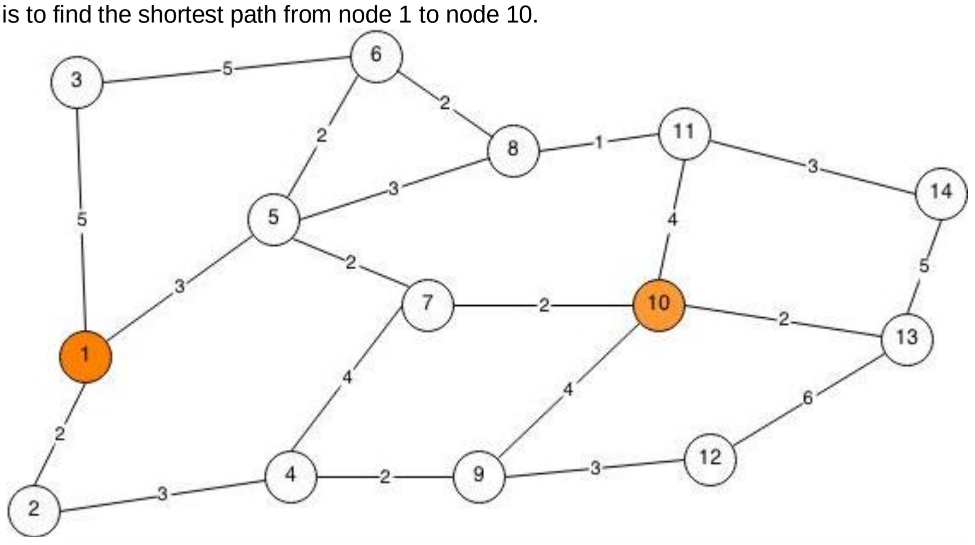
Consider a graph G, suppose we want to calculate the shortest path from node *a* to node *b* on graph G. Dijkstra's algorithm searches for the shortest path between nodes a and b by labelling nodes. Nodes can get a 'temporary' label and a 'permanent' label. To begin, node *a* is labelled with a permanent label of 0. All surrounding nodes, connected to node *a*, are labelled with a temporary label corresponding the weight of edge (for example distance) between node *a* and the particular node. All the other nodes in the graph have temporary labels of ∞. Dijkstra's algorithm chooses the node with the smallest temporary label and makes this label permanent.

Now we assume that node *i* is the (*k*+1)th node that has been given a permanent label. This means that node *i* is the *k*th closest node to node *a*. The temporary label of a remaining node *i'* , at this moment, is either the length of the shortest path between node *a* and node *i',* passing only through the *(k-1)* closest nodes to *a,* or ∞. Suppose the node with the lowest temporary label at this point, node *j*, is connected to node *i*, we replace it's temporary label by a permanent label, which is

$$\min\left\{\begin{array}{l} Current\ temporary\ label\ of\ node\ j \\ Permanent\ label\ node\ i + weight\ of\ edge\ (i,j)) \end{array}\right.$$

This process is continued until node *b* receives a permanent label. Once this is the case, the shortest path can be determined by working backwards from node *b* to node *a*. Every node *x* in between, that has a label of exactly the difference between the label of its successor (node *y)* and the edge weight of (*x,y),* is part of the shortest path.

To clarify the described method above, we perform Dijkstra's algorithm on the example network below. Our goal is to find the shortest path from node 1 to node 10.



At the start, we define the following labels (with * indicating a permanent label and the ith number representing the label of node i): [0* 2 5 ∞ 3 ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞]. Node 2 has the smallest temporary label. Therefore, node 2 gets a permanent label:

[0* 2* 5 ∞ 3 ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞]

This means that node 2 is the closest node to node 1. Now we can compute new temporary labels for all nodes connected to node 2. New temporary label node 4 = min{∞, 2+3} = 5. This makes node 5 the node with the smallest temporary label:

[0* 2* 5 5 3* ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞]

This enables us to recalculate temporary labels for nodes 6, 7 and 8, which results in 3 nodes with the lowest temporary labels:

[0* 2* 5 5 3* 5 5 6 ∞ ∞ ∞ ∞ ∞ ∞]

It does not matter whether we choose node 3, 4, 6 or 7 as the next node to give a permanent label (they all have a lowest temporary label of 5). We choose to give node 4 a permanent label (thereby changing temporary label of node 9):

[0* 2* 5 5* 3* 5 5 6 7 ∞ ∞ ∞ ∞ ∞]

Again, we can choose between multiple nodes. Now we choose to make node 6's label permanent (no other labels change):

[0* 2* 5 5* 3* 5* 5 6 7 ∞ ∞ ∞ ∞ ∞]

Next step is to make node 3'label permanent (again nothing else changes):

[0* 2* 5* 5* 3* 5* 5 6 7 ∞ ∞ ∞ ∞ ∞]

The last node with a temporary label of 5 is node 7, making it's label permanent leads to:

[0* 2* 5* 5* 3* 5* 5* 6 7 7 ∞ ∞ ∞ ∞]

Now, node 8 has the lowest temporary label. Making this node permanent leads to:

[0* 2* 5* 5* 3* 5* 5* 6* 7 7 7 ∞ ∞ ∞]

Now, we have 3 lowest temporary labels of 7. One of these temporary labels belongs to node 10, the destination of our shortest path. Therefore we choose to make node 10's label permanent, because then we know the length of the shortest path between node 1 and 10:

[0* 2* 5* 5* 3* 5* 5* 6* 7 7* 7 ∞ 9 ∞]

Once node 10 has a permanent label, we can determine the shortest path. We only have to compare permanent labels. The difference between permanent labels of nodes 10 and 7 = length of arc (7,10), so node 7 is on the shortest path. The difference between permanent labels of nodes 7 and 4 ≠ length of arc (4,7), so node 4 is **not** on the shortest path. The difference between permanent labels of nodes 7 and 5 = length of arc (5,7), so node 5 is on the shortest path. The difference between permanent labels of nodes 5 and 1 = length of arc (1,5), so node 1 is on the shortest path (of course). So the shortest path is 1-5-7-10, with a length of 7.