PUBLIC TRANSPORT ROUTE PLANNING WITH PREFERENCES

PUBLIC TRANSPORT ROUTE PLANNING WITH PREFERENCES

Thesis

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Chantal ECKHARDT
born in Vlaardingen, the Netherlands

Algorithmics Group
Department of Software Technology
Faculty EEMCS
Delft University of Technology
Delft, the Netherlands

Name: Chantal Eckhardt
Student number: 4235509
Defense date: 26-09-2017

ABSTRACT

Valys is social and recreational transport outside the region for people with mobility restrictions. Machine learning techniques are used to find patterns in data of Valys trips from last year (2016). With an 84% accuracy, it can be determined whether a Valys user will travel partly by train or not. Clustering shows that Valys users who travel partly by train can be split into two user groups: older users (age peak at 80 to 85 years) mostly without aiding tool and younger users (age peak at 45 to 50 years) mostly with mobility scooter or wheelchair as aiding tool.

The current journey planner of Valys takes a limited amount of train stations into account to drive to or from by taxi and cannot take user preferences into account. The purpose of this thesis is to find out how more Valys users can be attracted to travel partly by train and how the journey for Valys users that already travel partly by train can be improved. A new journey planner is designed to accomplish this and focuses on first one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode. All Pareto-optimal routes from source to target are computed with the Connection Scan Algorithm (CSA) by converting the unrestricted and the schedule-based transportation modes to connections. Every stop stores a list of non-dominated tuples, where each tuple contains the current values for the criteria. The following criteria are taken into account: travel time, number of transfers, frequency, walking distance, budget and cost.

The resulting algorithm is fast enough, but returns a large number of Pareto-optimal routes even after filtering out unreasonable routes. Next three methods to select a subset of representative routes are discussed and compared. As the weighted sum method achieves similar results to the fuzzy dominance method, but is much faster it is chosen as the best method (out of the three) to choose a subset of routes in the context of Valys. Allowing the user to depart 10 minutes earlier or later is found to be most suitable in travel time gain. The new journey planner spends on average 12.5 km more in budget and €0.5 more in cost, but gains on average 51.3 minutes in travel time and 1.1 in number of transfers compared to the current Valys journey planner.

Thesis committee:
Chair & university supervisor: Dr. Mathijs de Weerdt
Committee member: Dr. ir. Neil Yorke-Smith
Committee member: Dr. ir. Rob van Nes

PREFACE

The following overarching research question is answered in this thesis: how can more Valys users be attracted to travel partly by train and how can the journey for Valys users that already travel partly by train be improved. This thesis consist of three parts. Part I is a preliminary research that tries to get a better insight into the user group of Valys. Part II uses observations from part I to design and build a journey planner that takes user preferences into account. Both part I and II have their own introduction and conclusion, but have a joined future work and recommendation which is part III.

First I would like to thank Mathijs de Weerdt of the TU Delft for supervising me during my research. Next I would like to thank Stefan de Konink and Jasper Hartong of Calendar42 for their advisement about designing a new journey planner and to be able to use some of the tools of Calendar42. At last I would like to thank Transvision for providing the data of Valys trips from last year.

Chantal Eckhardt
Delft, the Netherlands
September 26, 2017

Contents

List of Figures

List of Tables

I

PRELIMINARY RESEARCH VALYS

1

INTRODUCTION

The goal of this part I is to get a better insight into the user group of Valys. Valys is social and recreational transport for people with mobility restrictions who travel more than five public transportation zones in the Netherlands. The public transportation zones in the Netherlands are displayed in figure 1.1. This part I first performs a general data analysis on the data of trips from last year (2016) which are delivered by Transvision [6]. Next machine learning is used to find patterns in this data. At last the human aspect is investigated by walking through the booking process at the Rotterdamse Mobiliteit Centrale and by holding a customer panel with real users. This research should answer the following questions:

- How can the users be generalized into different persona? What are the characteristics and limitations of every persona?
- Which factors are found to be important for each of these persona when planning the route from A to B?
- What are the reasons for a user to travel or not to travel partly by train? Which users can be attracted to start travelling partly by train?
- Can something really be improved about the current system?

1

Figure 1.1: Public transportation zones in The Netherlands [1]

A grouping into personas is successful when every personas can be distinguished from another personas by his or her travel behaviour or travel needs. Valys consists of three travel types: Valys Basis, Valys Begeleid and Valys Vrij. Users from Valys Basis only travel by taxi and users from Valys Begeleid and Valys Vrij travel partly by train and partly by taxi. Valys Begeleid offers an assistance at 31 train stations in the Netherlands to get to the right platform and into the train. It is possible that every one of these travel types are done by people with very distinctive travel behaviour or travel needs and that this will lead to three different personas.

At the beginning of the year the *Persoonlijk Kilometer Budget* (PKB) of the Valys user is set again to its initial value. For most users this is 600 km and for users with a high PKB this is 2250 km. One can qualify to get a high PKB budget when he or she cannot travel by train due to a medical condition or other restriction. The user only loses kilometers from his or her budget when using the taxi. When the user completely uses his or her PKB travelling by taxi becomes more expensive. This means more trips for the same amount of money are possible when travelling partly by train (Valys Begeleid or Valys Vrij). When a new user joins Valys later in the year he or she will have a kilometer budget related to the amount of months left.

The outline of this part I is as follows. Chapter 2 performs a general data analysis on the data of Valys trips from last year. Chapter 3 uses machine learning to find patterns in the data. Next chapter 4 investigates the human aspect. At last chapter 5 gives a conclusion about the found insights about the Valys user.

2

GENERAL DATA ANALYSIS

When a user books a trip online or over the phone the data of that trip is saved. As a result there is a lot of data available about the user and the trip itself. Section 2.1 gives a general view of the Valys user and the made trips. Section 2.2 compares the travel time of trips partly by train with trips not partly by train. This is all done for trips from last year (2016). The goal of this chapter is to give a first impression about the Valys user and the difference between trips partly and not partly by train.

2.1. THE USER AND THE MADE TRIPS

The stereotype Valys user is a grandpa or a grandma who wants to visit his or her children. This is why it is no surprise that last year most trips were made by people aged 84 years as is deducible from figure 2.1. This age is the age the user was on the day of the trip. The age averaged over the trips is equal to 70 years. Most of the trips from last year were made by female users (73%). Most Valys users (54%) make only 1 to 4 trips a year, 26% makes 4 to 8 trips and the rest makes more than 8 trips a year.

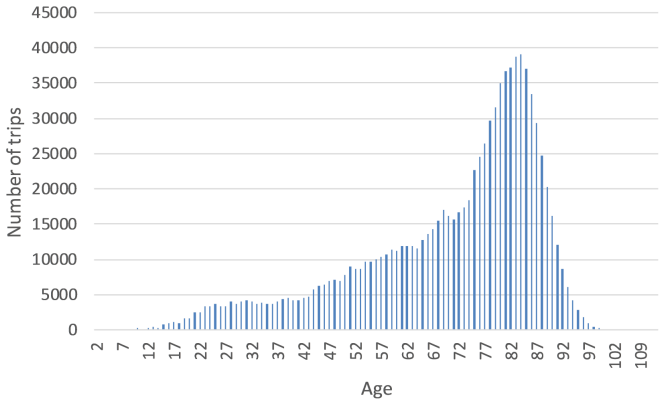


Figure 2.1: Age distribution of the Valys users

December is the peak month of the year in the number of trips and the reason for this is that a lot of users visit their family for Christmas. Most trips occur during the weekend and this is probably due to children of the users not having to work during the weekend. The peak hour of departure is around 9/10 o'clock and not many trips are made during peak hours. A precise distribution of the trips over the months, days and hours can be found in Appendix A.

For 40% of the trips last year the user brought an aiding tool with them. The walker and the wheelchair are the most common used aiding tools for all Valys types. For 29% of the trips the user had a client indication. Visual impairment and impaired hearing are the most common client indications. A precise distribution of the aiding tools and client indications can be found in Appendix A.

2.2. COMPARING VALYS TYPES

This section compares the Valys trips partly and not partly by train on the number of trips, the average trip distance and travel time. Of the users that travel partly by train 66% had a taxi to the station and from the station, 30% had a taxi to the station or from the station and 3% used only the train. When counting the number of trips of Valys Begeleid, Valys Vrij, taxi to the station (Valys Basis) or taxi from the station (Valys Basis) this covered only 2.5% of the total amount of trips. This means 97.5% of the trips were done completely by taxi.

The average distance by train for a Valys Begeleid or a Valys Vrij trip is 95 km (113 minutes) and the average distance by taxi for a Valys trip is 52 km (85 minutes). It is expected given the average distance by taxi and the average amount of trips per user per year that most users do not spend their PKB. This is correct as only 5% of the users that did a trip with Valys Basis last year and 14% of the users that did a trip with Valys Begeleid or Valys Vrij last year completely spend their budget.

Now the travel time of trips partly by train (Valys Begeleid and Valys Vrij) is compared with the travel time of trips completely by taxi (Valys Basis). The expectation is that a trip

partly by train is longer than the equivalent not partly by train and that this is caused by the taxi not directly going to the end destination and the planned in waiting time by Valys.

For every departure–arrival zipcode combination where there exist a version partly by train and a version completely by taxi in the data the difference in travel time is computed. This leaves just over 2000 zipcode combinations (the letters of the zip code are left out). Figure 2.2 displays the number of occurrences per difference in travel time, where a positive difference means it took longer to travel by train than completely by taxi. As expected a trip partly by train is longer then the equivalent not partly by train, most trips partly by train are between 12 to 84 minutes longer than trips completely by taxi.

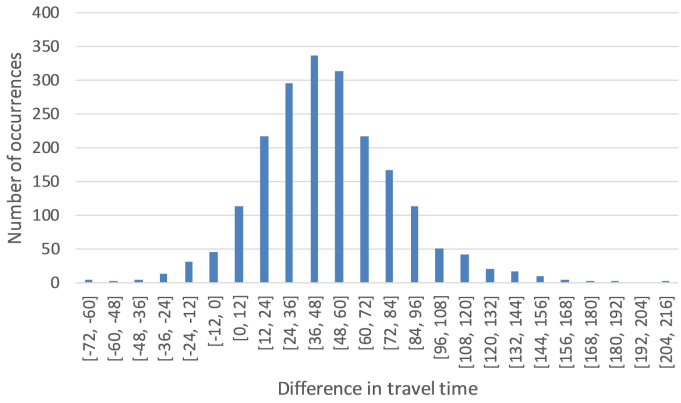


Figure 2.2: Number of occurrences per difference in travel time between trips completely by taxi and trips partly by train. A positive difference means it took longer to go partly by train than completely by taxi.

One of the reasons it sometimes takes longer to travel partly by train becomes clear with these two examples. In the first example it takes 60 minutes to go from an address in Vlaardingen to another address in Amsterdam by only taxi. The equivalent trip partly by train takes 130 minutes. When splitting this up in train and taxi the arrival and departure times are: 1811-1818 (taxi) and 1843-2021 (train). This means 25 minutes in total are spent walking to the platform and waiting for the train. The minimum waiting time between transfers is 15 minutes in the current system and the other 10 minutes are probably needed to wait for the next train. In the second example it takes 54 minutes to go from an address in Woerden to an address in Boxtel by only taxi. The equivalent trip partly by train takes 116 minutes. When splitting this up in train and taxi the arrival and departure times are: 1856-1920 (taxi), 1953-2020 (train) and 2034-2052 (taxi). This means 47 minutes in total are spent waiting for and walking to the train or taxi.

Both these examples indicate a lot of time is reserved for walking and waiting and this is one of the reasons it takes (most of the time) longer to travel partly by train. Not every user needs this much reserved time and this is something that can be improved in the current system. The next chapter uses machine learning to find patterns in the data

of the Valys trips from last year (2016).

3

MACHINE LEARNING

This chapter uses machine learning to find patterns in the data of the Valys trips from last year (2016). This is done to get a better insight about the Valys user and to see if there are users with similar characteristics that show similar behaviour, which could then be used to make future predictions. Machine learning is used to find out which criteria of the trip or the user can predict: when someone will travel partly by train or not (section 3.3), the chosen Valys type (section 3.4) and when someones will make a transfer or not (section 3.5). Data of trips from last year (2016) are used. It is possible that the data is not enough to make such predictions. Section 3.1 describes the used variables, section 3.2 gives expectations for the classifications of section 3.3-3.5 and section 3.6 tries to divide some of the Valys users into clusters.

Data of the trips are divided into classes to make predictions. To predict if a user will travel partly by train the trips are divided into two classes: train use or no train use (Valys Begeleid and Valys Vrij are seen as one class). To predict the Valys type the data is divided into three classes: Valys Basis, Valys Begeleid and Valys Vrij. To predict if someone will make a train transfer while travelling partly by train the data is divided into two classes: transfer or no transfer.

For each division into classes a part of the data is used to train on a classifier and another part of the data is used to test this classifier. One classifier is considered to be better than another classifier when the number of misclassified objects of the test set is lower. Table 3.1 displays a list of classifiers and their abbreviations that are used in this chapter and this list is conducted by choosing classifiers of different types. In the next sections only the abbreviations are used. The Matlab Pattern Recognition Toolbox (PRTools) is used for classification [2].

Table 3.1: List of classifiers and their abbreviation

Classifier	Abbrevation	Type
Nearest Mean Classifier	nmc	Linear
Linear Bayes Normal Classifier	ldc	Linear
Fisher's Least Square Linear Discriminant Classifier	fisherc	Linear
Logistic linear classifier	loglc	Linear
Quadratic Bayes Normal Classifier	qdc	Quadratic
Parzen classifier	parzenc	Parzen
Back-propagation trained feed-forward neural network classifier	bpxnc	Neural network
Levenberg-Marquardt trained feed-forward neural network classifier	lmnc	Neural network
Decision tree classifier	treec	Decision tree

3.1. THE DATA

The data of the trips from last year (2016) are used to build a classification system. Only criteria that are known about the user or the trip on the moment of the booking are taken into account. Criteria about the user include:

- Age
- Gender
- If AVG (assistance) would be required when travelling by train
- If the user requested to be guaranteed to arrive at a certain time
- Initial balance and end balance of the budget of the previous year (2015)
- Used aiding tools and client indication (e.g. bad hearing)
- Amount of luggage
- Number of passengers (it is possible to bring someone along)
- Number of trips partly by train made by the user last year (2015)

The following criteria are taken into account per trip

- The zipcode of the location of departure and the location of arrival
- Total distance
- If the booking is made by phone or by internet
- Month, weekday and hour of the day of booking and the day of departure
- Amount of rain and temperature on the day of the trip in the town Bilt. The Bilt is chosen as location because of its central location in the Netherlands.

- If the sun was down during a part of the trip

The last two criteria of the trip are not provided by Transvision, but are from the KNMI [7]. Trips that were booked but did not actually took place are filtered out of the data. Aiding tools are categorized, for example 8 different types of wheelchairs exist in the data and they are seen as one type of aiding tool. Every criteria is normalized between 0 and 1000.

3.2. EXPECTATIONS

Before using machine learning to find patterns in the data, one could have expectations about the outcome and the possible influence of some criteria. A series of classifiers are put to the test, depending on the spread of the criteria some classifiers will be better than others.

The current expectations are as follows. A user is less likely to travel partly by train when it is cold, when in need of AVG, when a lot of baggage is brought on the trip or when the user is of really old age. A user is more likely to travel partly by train when he or she makes a lot of trips per year (to save up budget), books the trip by internet instead of the phone and already made a few train trips last year. Predicting whether a trip is Valys Begeleid or Valys Vrij trip will be hard. Possibly the only difference is in the used aiding tool and if AVG is required. The same accounts for predicting whether someone will make a train transfer or not.

3.3. CLASSIFICATION ON TRAIN USE

The data is divided into two classes: train use or no train use. The goal of this section is to find out which criteria of the trip or user can predict if someone will travel partly by train or not. For the first test a dataset is used that consists of all trips done partly by train and an even amount of trips not done by train. This means only a selection of Valys Basis trips are used, since in reality a lot more Valys Basis trips are done than Valys Begeleid and Valys Vrij combined. Of this dataset 30% is used for training and 70% for testing on different classifiers. By using such as big part of the dataset for testing the resulting amount of misclassified objects is accurate. The used method is a non-exhaustive cross validation method, since the computation error is not computed for all possible training and testing set combinations. This is good enough in this case, since knowing which criteria are of influence is more important than the classification error itself.

For the first test different classifiers are trained on train use and no train use. Table 3.2 shows the average classification error per classifier on the test set (based on 5 runs). This classification error is the number of misclassified trips divided by the total number of trips. The second column of table 3.2 shows the results when no feature extraction is performed on the data, the third when a fisher mapping is performed on the data and the fourth column when a Principal Component Analysis (PCA) with dimensionality reduction to 22 is performed on the data. The dimensionality reduction to 22 is based on what gave the lowest classification error. Both fisher mapping and PCA are feature extraction methods that can influence the classification error. For the *treec* a pruning value of 15 is used and this amount of pruning is based on what gave the lowest classification error.

The fisher mapping decreased the classification error for some classifiers and PCA increased the classification error for some classifiers. The results in table 3.2 indicate that training the data on the *loglc* or the *treec* gives the lowest error. An error of 0.16 means that 84% of the trips can be classified correctly on train use or no train use by this classifier. The confusion matrix for the *loglc* is displayed in table 3.3 and the confusion matrix for the *treec* is displayed in table 3.4. Most misclassified trips of the *loglc* classifier were predicted to use the train, but in reality did not and for the *treec* it was the other way around.

Table 3.2: Average classification error per classifier when training on train use or no train use. Equal number of trips for train use and no train use.

	No feature extraction	Fisher mapping	Principal Component Analysis (d=22)
nmc	0.33	0.18	0.33
ldc	0.18	0.18	0.18
fisherc	0.18	0.18	0.18
loglc	0.16	0.17	0.16
qdc	0.27	0.18	0.2
parzenc	0.3	0.17	0.3
bpxnc	0.18	0.17	0.17
lmnc	0.18	0.17	0.17
treec (p=15)	0.16	0.17	0.24

Table 3.3: Confusion matrix for the loglc classifier when training on train use and no train use. Equal number of trips for train use and no train use.

Actual \ Predicted	No train use	Train use
No train use	4600 (81.4%)	1051 (18.6%)
Train use	667 (11.8%)	4984 (88.2%)

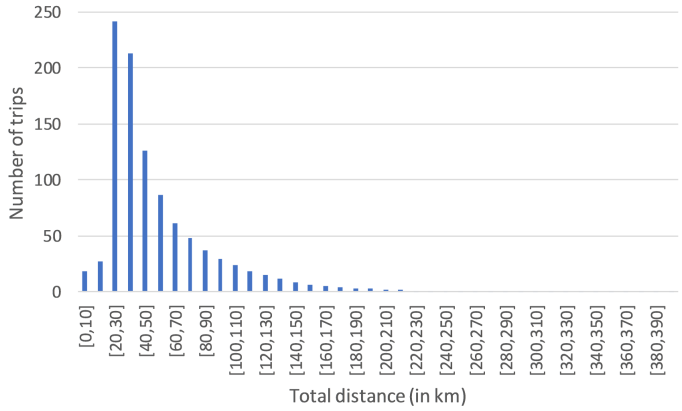
Table 3.4: Confusion matrix for the treec (p=15) classifier when training on train use and no train use. Equal number of trips for train use and no train use.

Actual \ Predicted	No train use	Train use
No train use	4951 (87.6)%	700 (12.4%)
Train use	970 (17.2%)	4681 (82.8%)

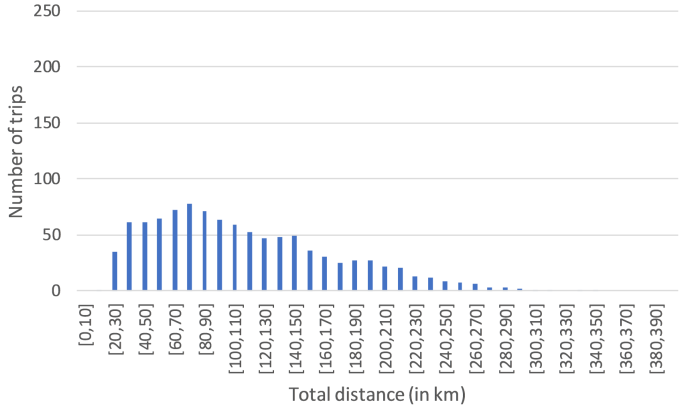
Now the question remains which criteria decide whether a trip is partly by train or not. The classification error when only one variable is used, is computed for the *loglc* and the *treec*. The results are displayed in table 3.5.

Table 3.5: Classification error for training on train use or no train use when only a single criterion is taken into account. Only criteria with an error below 0.4 are displayed. The classifier that gave the lowest classification error is in parentheses.

Criteria	Classification error
Total distance	0.25 (treec)
Number of train trips in 2015	0.26 (loglc)
Budget at the end of 2015	0.38 (loglc/treec)
Age	0.39 (treec)



(a) No train use



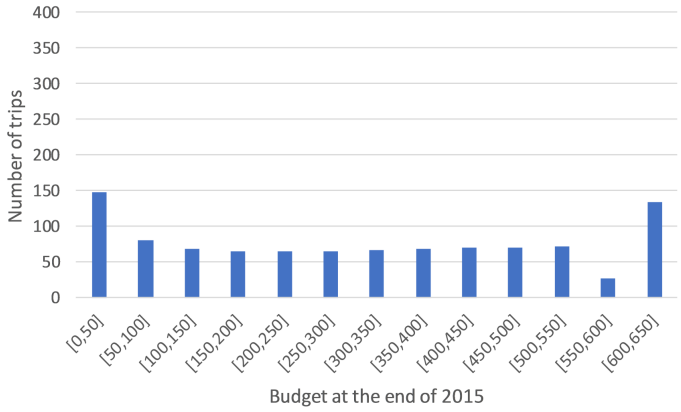
(b) Train use

Figure 3.1: Total distance distribution for train use and no train use. The total number of trips is normalized to 1000.

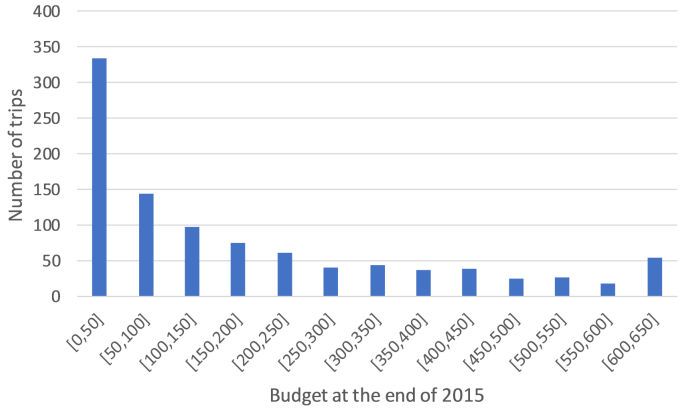
The criterion with the most individual influence is the total distance. The total dis-

tance distribution for train use and no train use is displayed in figure 3.1. For trips not partly by train, mostly a distance between 20 and 70 km is covered and trips that use the train are divided between a bigger range of covered km. Trips where a distance lower than 10.9 km or higher than 340.1 km is covered are never partly by train. In total 36031 trips of 2016 fall into this category.

The criterion with the second most individual influence is the number of train trips in 2015. This is a very logical influence, since it is more likely for a user to travel partly by train when he or she has done this before in the past. Next the third criterion with the most individual influence is the budget at the end of 2015. Figure 3.2 shows the budget at the end of 2015 distribution for train users and non-train users and this shows that train users are more likely to use most of their budget when being compared to non-train users.



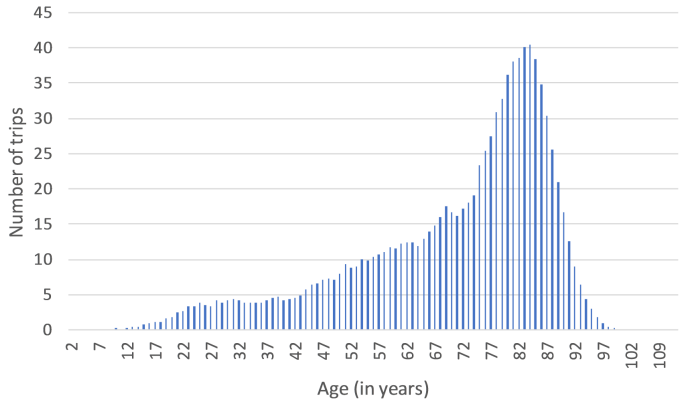
(a) No train use



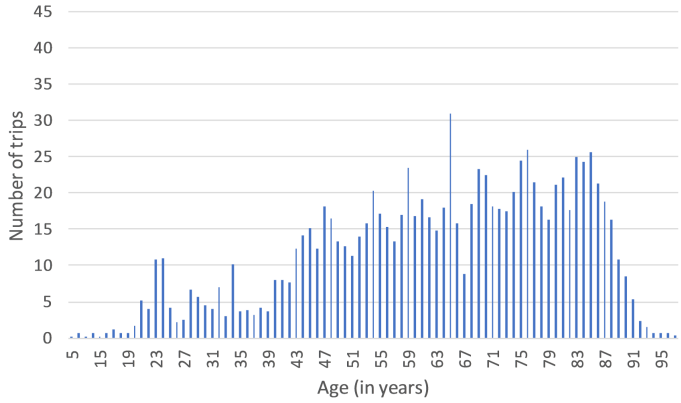
(b) Train use

Figure 3.2: Budget at the end of 2015 distribution for train use and no train use. Budget 650 or above is not shown. The total number of trips is normalized to 1000.

The last criterion with an individual classification error below 0.4 is the age. The age distribution for train use and no train use is displayed in figure 3.3. Most trips where the train is not used the user is between 60 and 90 years and for no train use the common age range is larger. The peak for no train use is 84 years and the peak for train use is at 65 years. The oldest user to travel partly by train is 97 and for no train use the oldest user is a bit older.



(a) No train use



(b) Train use

Figure 3.3: Age distribution for train use and no train use. The total number of trips is normalized to 1000.

Table 3.6 shows the four best combinations of two criteria based on classification error when using the *loglc* or the *treec*. The two criteria with the most individual influence are also the combination of two with the lowest classification error. Figure 3.4 displays these two criteria for a training set. The blue points represent train use and the red points represent no train use. Especially the number of trips partly by train in 2015 gives a clear distinction between train and no train use. The second best combination combines the budget at the beginning of the year with the number of train trips in 2015. This combination makes the high PKB users who have more budget (> 600 km) and cannot travel partly by train due to their condition more distinctive. The other two combinations of criteria in table 3.6 have only the slightest effect when comparing it to the classification error of only the total distance.

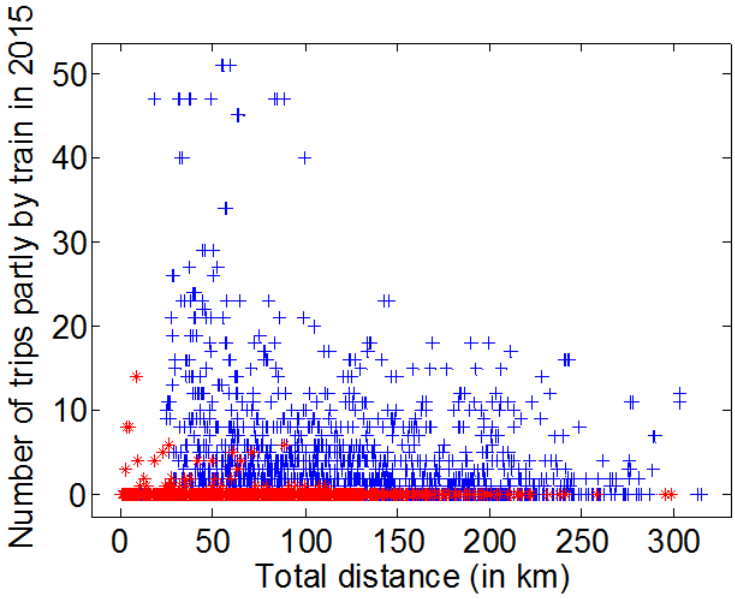


Figure 3.4: Number of trips partly by train in 2015 vs total distance for a training set. The red points represent no train use and the blue points represent train use. The graph is generated by PRTools [2]

Table 3.6: The four best combinations of two criteria based on their classification error. The classifier that gave the lowest classification error is in parentheses.

Criteria	Classification error
Total distance, number of train trips in 2015	0.18 (treec)
Budget at the beginning of 2015, number of train trips in 2015	0.23 (treec)
Total distance, client indication	0.24 (treec)
Total distance, budget at the end of 2015	0.24 (treec)

When building another classification system where the classes are divided in: misclassified objects and good classified objects (based on train use), then the resulting classification system does not work well enough. This means it is not possible with the current data to indicate which users will probably be classified into the wrong class beforehand.

The amount of baggage, if AVG is required and if the booking is made by phone or internet are variables that were expected to have an influence on whether someone would want to travel partly by train or not, but in reality these variables were not strong enough to give such an indication. On the other hand age, budget and the number of train trips in 2015 are as expected variables that had an influence on whether someone would travel partly by train or not. The total distance was not considered to be influential, but when considering that more budget is needed for longer trips it seems reasonable that smaller trips are more common not partly by train.

Appendix B.1 shows the classification results when the trips are filtered on trips with a total distance between 30 and 90 km, this is done to limit the power of the total distance variable. Appendix B.2 shows the classification results when the trips are filtered on trips done by users who participated in both trips partly and not partly by train to find out if there is decisive factor that influences the decision to travel partly by train or not. The next section tries to see if there are criteria that can predict the chosen Valys type.

3.4. CLASSIFICATION ON VALYS TYPE

The goal of this section is to find out if it is possible to predict the Valys type a Valys user chooses. The data is divided into three classes: Valys Basis, Valys Begeleid and Valys Vrij. The data is trained and tested just as in the previous sections and the average classification results are displayed in table 3.7. The lowest classification error of 0.35 (*loglc*) is still quite high. When looking at the corresponding confusion matrix in table 3.8 it seems the classifier is having a hard time distinguishing Valys Begeleid and Begeleid Vrij trips. This is as expected.

Table 3.7: Average classification error per classifier when training on the Valys type. Equal number of trips for Valys Basis, Valys Begeleid and Valys Vrij.

	No feature extraction	Fisher mapping	Principal Component Analysis (d=22)
nmc	0.51	0.36	0.51
ldc	0.37	0.36	0.37
fisherc	0.46	0.48	0.48
loglc	0.35	0.37	0.36
qdc	0.48	0.37	0.38
parzenc	0.46	0.36	0.47
bpxnc	0.4	0.36	0.37
lmnc	0.37	0.37	0.36
treec (p=15)	0.37	0.37	0.43

Table 3.8: Confusion matrix for the loglc when training on the Valys type. Equal number of trips for Valys Basis. Valys Begeleid and Valys Vrij.

Actual \ Predicted	Valys Basis	Valys Begeleid	Valys Vrij
Valys Basis	1977 (83.4%)	184 (7.8%)	210 (8.9%)
Valys Begeleid	320 (13.5%)	1511 (63.7%)	540 (22.8%)
Valys Vrij	411 (17.3%)	794 (33.5%)	1166 (49.2%)

3.5. CLASSIFICATION ON TRANSFERRING

The goal is to find out which criteria (besides the total distance) have an influence on whether someone would want to make a transfer or not. The data is first filtered on trips partly by train. This data is divided into two classes: train transfer and no train transfer.

Figure 3.9 displays the classification results and shows that the lowest classification error is 0.33.

Table 3.9: Average classification error per classifier when training on train transfer or no train transfer. Equal amount of trips for transfer and no transfer.

	No feature extraction	Fisher mapping	Principal Component Analysis (d=22)
nmc	0.39	0.34	0.39
ldc	0.33	0.34	0.34
fisherc	0.33	0.34	0.34
loglc	0.33	0.34	0.34
qdc	0.39	0.34	0.36
parzenc	0.37	0.34	0.37
bpxnc	0.36	0.35	0.34
lmnc	0.39	0.37	0.38
treec (p=15)	0.33	0.34	0.38

When looking at the individual influence of the criteria in table 3.10 only a few new criteria have an influence. Besides the total distance, the used aiding tool and if this aiding tool required assistance criteria have a small individual influence. Figure 3.5 compares the need of AVG for train trips with or without a transfer.

Surprisingly of the trips where the user made a train transfer more users in terms of percentage had an AVG aiding tool with them compared to the trips where the user did not make a train transfer. Also for trips where the user made a train transfer less users in terms of percentage had no aiding tool compared to trips where the user did not make a transfer. From this it can not be deduced which people do or do not make a transfer, but it does say that using a AVG aiding tool does not prevent one from making a transfer. The next section performs a clustering on a subset of the trips of 2016.

Table 3.10: Classification error for individual criteria that have an error below 0.4 on the treec. Classification is on train transfer or no train transfer.

Criteria	Classification error
Total distance	0.340
If AVG (assistance) would be required when travelling by train	0.397
Aiding tool	0.399

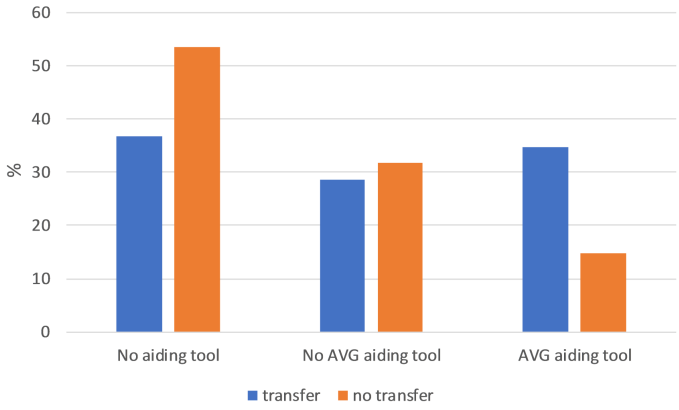


Figure 3.5: Distribution for no aiding tool, no AVG aiding tool and AVG aiding tool for train trips with or without a transfer.

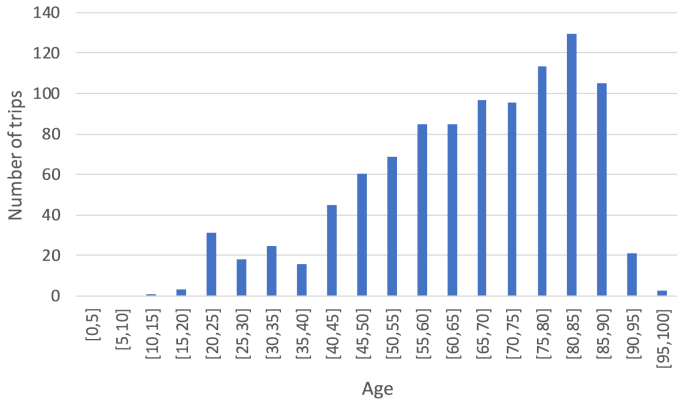
3.6. CLUSTERING

A clustering is performed on two different data sets. The first data set consists of trips done partly by train in 2016. The second data set consists of trips with a travel distance larger than 10 km not done partly by train of which the involved user spend most of his or her budget (less than or 50 km left at the end of the year) and did not participate in trips partly by train in 2016. This last data set consists of trips of which the user could be convinced to start travelling partly by train, based on the distance of the trip and the amount of budget left of the user.

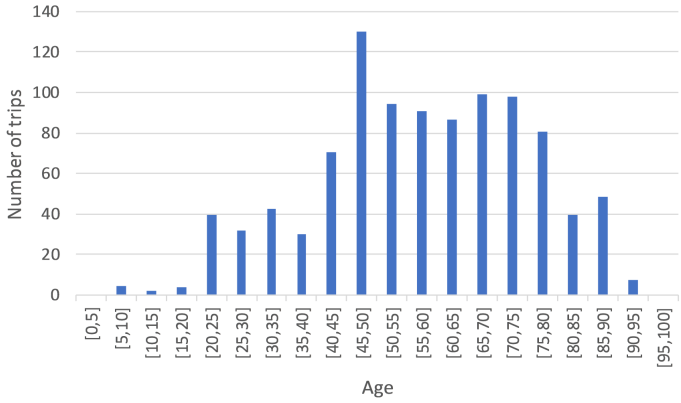
Clustering is an unsupervised machine learning method, meaning it tries to describe (hidden) structures from unlabeled data. A clustering in for example two groups can be called successful if both groups are distinctive enough in their features. The goal of this section is to find persona in the data and to find the maximum increase possible in trips partly by train.

3.6.1. TRIPS PARTLY BY TRAIN

First it is tried to divide the data of trips partly by train into two clusters. The clustering method that gives the most distinctive groups is hierarchical clustering with complete linkage. When comparing the two clusters, two criteria are very distinctive per cluster and this is visible in figure 3.6 and 3.7. The first cluster consists of users with an age peak at 80 to 85 years of which most users brought no aiding tool and the second cluster consists of users with an age peak at 45 to 50 years of which most users brought a wheelchair or mobility scooter as aiding tool. The first cluster is more than three times bigger than the second cluster in the number of trips (6330 vs 1744).



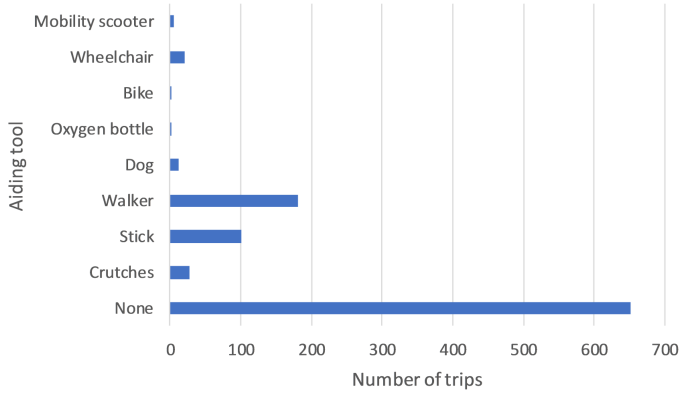
(a) Cluster 1



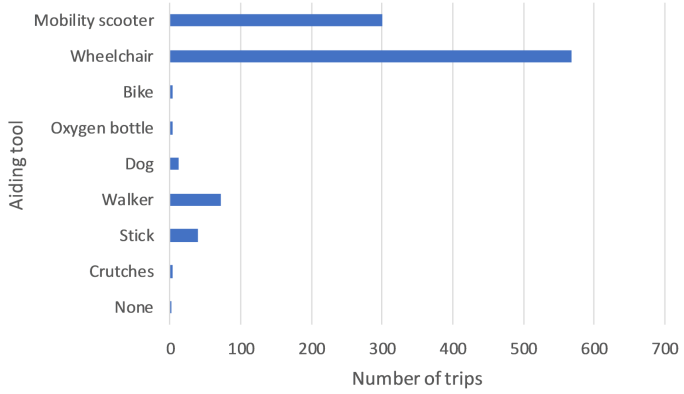
(b) Cluster 2

Figure 3.6: Age distribution for cluster 1 and cluster 2. Clustering is performed on trips partly by train. The total number of trips is normalized to 1000.

When dividing the data set into three clusters the biggest cluster from the previous clustering (cluster 1) is now divided into two new clusters (let's call them 1a and 1b). The difference between these two clusters is that the users from cluster 1a make on average 4.2 trips per year and the users from 1b make on average 2.5 trips per year. These trips are equally divided over the year for cluster 1a, but for cluster 1b not many trips occurred during the first three months of the year. Dividing into more clusters gave not very distinctive clusters, this means that the users cannot be divided into more groups that have clear behavioural differences.



(a) Cluster 1



(b) Cluster 2

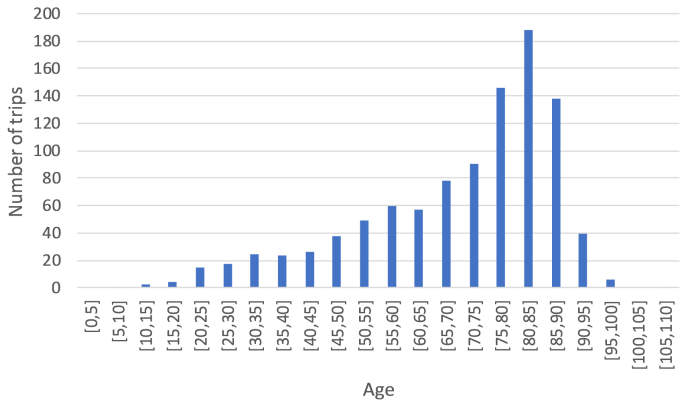
Figure 3.7: Aiding tool distribution for cluster 1 and cluster 2. Clustering is performed on trips partly by train. The total number of trips is normalized to 1000.

3.6.2. TRIPS NOT PARTLY BY TRAIN

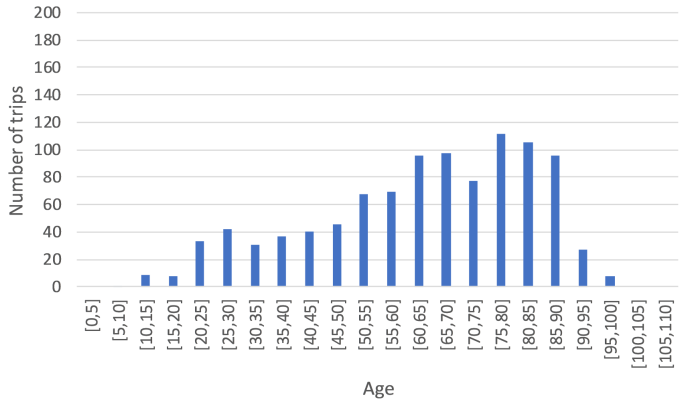
This section tries to cluster the trips not partly by train of which the user could be convinced to start travelling by train. Since the chosen subset of trips partly by train is very big (222760 objects), only 10000 samples are used for the clustering to lower the running time and the memory usage. These samples are randomly chosen.

First the data is divided into two clusters with the same method as in the previous section and the two most distinctive criteria are displayed in figure 3.8 and 3.9. The first cluster consists again of users with an age peak at 80 to 85 years of which most users brought no aiding tool. The second cluster consists again of users with a mobility scooter or wheelchair as aiding tool, however the age is spread over a bigger range (between 60 and 90). Another feature that is distinctive between these two clusters is the number of days booked ahead. The users of the first cluster book on average 3.3 days ahead and

the users from the second cluster book on average 6.3 days ahead. This means that users with a mobility scooter or wheelchair as aiding tool book on average three more days ahead than users without aiding tool.



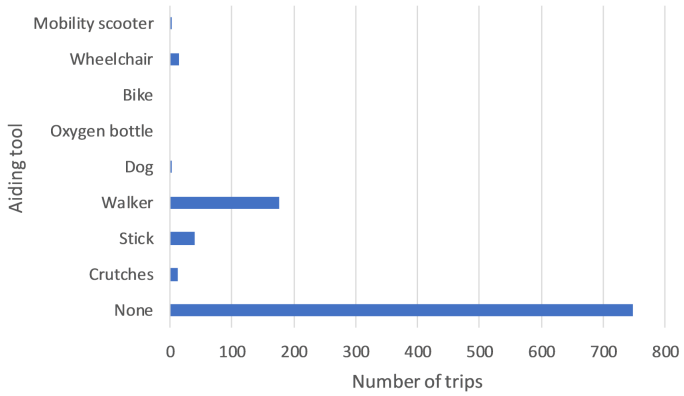
(a) Cluster 1



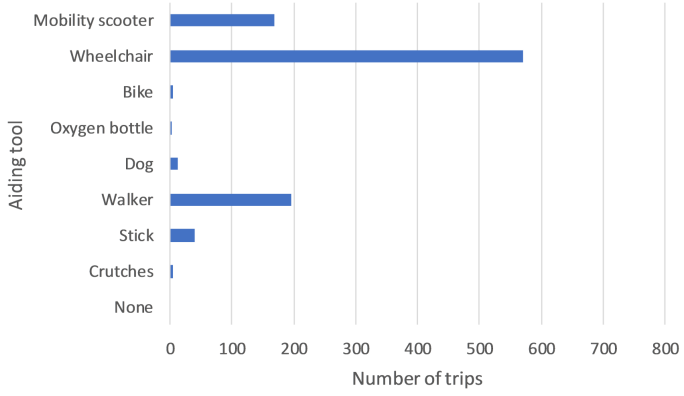
(b) Cluster 2

Figure 3.8: Age distribution for cluster 1 and cluster 2. Clustering is performed on trips not partly by train. The total number of trips is normalized to 1000.

Dividing into more clusters does not give very distinctive clusters, again this means that the users cannot be divided into more groups that have clear differences. In conclusion the found personas in the trips partly by train and not partly by train are very similar. The biggest difference is that the age group 40 to 60 is represented more by trips partly by train. This indicates again that age is an important factor in predicting whether someone travels partly by train.



(a) Cluster 1



(b) Cluster 2

Figure 3.9: Aiding tool distribution for cluster 1 and cluster 2. Clustering is performed on trips not partly by train. The total number of trips is normalized to 1000.

3.6.3. MAXIMUM INCREASE IN TRIPS PARTLY BY TRAIN

Now the question remains what is the maximum increase in trips partly by train. In the previous section the size of the subset of trips not partly by train was 222760. When also leaving out trips below 20 km in distance and trips done by users above 80 years 144050 trips remain. The assumption is that older users have a lower change of starting to travel partly by train. When being even stricter with age, leaving out users above 70 years, the number of trips decreases to 92881. This is 9.6% of the total number of trips in 2016. Still some of these users could have reasons not to travel partly by train. Reasons for this include they are really not able to (medical information is not available), the budget is exactly enough and they are not interested in making more trips with Valys or they are just to stubborn in not wanting to travel partly by train.

Some trips are never done partly by train. Trips below 10 km can almost never be

done partly by train. Users above 85 years, users who have more than 200 km in budget left or PKB users have a really low change and/or incentive of starting to travel partly by train. Of the total number of trips, 57.6% fall into this category and this means that more than half of the trips will probably never be done partly by train. The next chapter looks at the human aspect behind the data and uses a customer panel to discover information that can not be retrieved from the data.

3.7. SUMMARY

This chapter uses machine learning to find patterns in the data of the Valys trips from last year (2016). Of the Valys trips 84% can be classified correctly on train use or no train use by the *loglc* or *treec*. The most influential criteria on train use are: total distance, number of train trips in 2015, budget at the end of 2015 and age. Of the Valys trips only 65% can be classified correctly on Valys type. All classifiers have a hard time distinguishing Valys Begeleid and Valys Vrij trips. Of the Valys trips partly by train 67% can be classified correctly on the use of a train transfer. The most influential criteria for transferring are: total distance, if AVG (assistance) would be required when travelling by train and the aiding tool.

Clustering on two different data sets is performed. Clustering the trips partly by train results in two distinctive clusters. The fist cluster consists of users with an age peak at 80 to 85 years of which most users brought no aiding tool. The second cluster consists of users with an age peak at 45 to 50 years of which most users brought a wheelchair or mobility scooter as aiding tool. Clustering the trips not partly by train of which users could be convinced to start travelling by train results in two distinctive clusters. The first cluster consists again of users with an age peak at 80 to 85 years of which most users brought no aiding tool. The second cluster consists again of users with an mobility scooter or wheelchair as aiding tool, however the age is spread over a bigger range (between 60 and 90). The maximum increase in trips partly by train is approximated at 9.6% of the total number of trips.

4

HUMAN ASPECT

This chapter investigates the human aspect of travelling with Valys. The goal is to gain more knowledge about why some users do not travel partly by train and how the journey for users who already travel partly by train can be improved. Section 4.1 summarizes the observations gathered at the callcenter and section 4.2 summarizes the observations gathered from a customer panel.

4.1. ROTTERDAMSE MOBILITEIT CENTRALE (RMC)

Since 92% of the bookings from last year were made by phone this is an important part of the booking process. The RMC is visited to get a better understanding of the multi-modal travel behaviour of the Valys user. The following observations are based on users calling to make a booking and interviewing callcenter staff.

4.1.1. THE USER

Multiple reasons were mentioned why users use the transport of Valys: to go to family, a cremation, a holiday resort, a cruise, the hospital or a day out (e.g. the beach). It is interesting to note that most users (average age 70) make the bookings themselves by phone and most have the intention of booking a trip for the day after. Frequently asked questions of the users were: how long does it take to go from A to B? How much kilometer budget do I have left after this trip? Can I still use the kilometers I have left from last year?

Apparently some users complain about illogical taxi routes or long taxi rides. Other users already looked online which train they could take or are happy when someone goes with them in the train. Since there is most of the time no difference between the cheapest, fastest and best route for Valys it seems irrelevant that this is listed on the screen. The callcenter staff has to ask every time when a user wants to book a trip whether they would like to travel partly by train. Different reasons were given why people do not want to travel partly by train:

- I find it not easy to get into the train

- It is too cold outside or too dark outside
- I find transferring from taxi to train scary
- I have a lot of kilometer budget left
- It is a long ride, so I prefer the taxi
- I just do not want to, I am afraid

4.1.2. DATA ANALYSIS

The following assumptions were made by the callcenter staff and are checked by data analyses.

- Most users book their trip a day before. This is correct: 10% book the same day, 43% book a day before, 17% book two days before and other users book more than 2 days before.
- When the weather is better more users travel partly by train instead of completely by taxi. Last year (2016) the warmest months in the Netherlands were July, August and September and the coldest months were December, January and February. When comparing the amount of trips for these months that go partly by train it seems that in the warmer months 29% more trips occurred partly by train than in the colder months.
- When users have a low kilometer budget they are more inclined to go partly by train. Figure 4.1 displays the number of trips per months filtered on trips that go partly by train and users who completely used their kilometer budget. This figure indicates there is a correlation between the kilometer budget and the amount of trips made at the beginning and the amount of trips made at the end of the year.

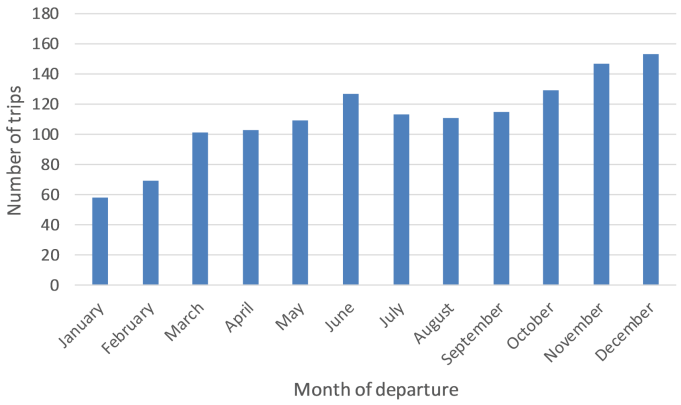


Figure 4.1: Number of trips per months filtered on trips that go partly by train and users who completely used their kilometer budget

4.2. CUSTOMER PANEL

A customer panel was held to find out from Valys users what can be changed about the route planning to improve their (train) journey or what can be changed about the route planning to convince non-train users to start travelling partly by train. This section gives a summary of the most important observations related to route planning, a summary of all observations can be found in Appendix C.

Most users present at the customer panel indicated that a budget of 600 km was too low for their needs. The main incentive for users to travel partly by train is to be able to make more trips for the same amount of money, since users only spend a part of their budget when travelling by taxi and do not spend their budget when travelling by train. When a user completely spends his or her budget the price per km for the taxi increases from €0.20 to €1.31. How much budget the trips costs and the real cost itself are important factors for most users.

One user indicated to still be able to drive, but that the walking distance from the parking lot to the destination was a problem and that Valys was the solution to limit the walking distance. Some users travel with Valys Begeleid just because the travel time is shorter than for Valys Vrij, the reason for this is that Valys Begeleid only drives by taxi to the bigger train stations. Someone's age or aiding tool does not seem to give a direct preference in routing. It seems to be a personal matter if someone prefers for example a train ride with one transfer or a train ride with no transfers which is a half hour longer. Transferring is considered to be exciting and stressful for some users, but cosy for others. Some users indicated to want certainty about the train rides and this could be related to the frequency of the train ride.

A couple of reasons were mentioned why some user do not use the public transport. Two of which could be solved with a new journey planner: not wanting to transfer and finding the travel time too long when travelling partly by train. The needed average transfer time of the questioned users ranged between 3 to 15 minutes, indicating the user should be able to adjust this to his or her own preference.

The Valys users present a the customer panel seem to want a more personalized journey planner. In summary the following criteria are found to be important by the users: travel time, number of transfers, frequency, walking distance, budget and cost. Mentioned restrictions were maximum walking distance, walking speed and maximum number of transfers.

5

Conclusion

The Valys users that travel partly by train can be generalized into two different personas: users without aiding tool with an age peak at 80 to 85 and users with aiding tool (mostly wheelchair or mobility scooter) with an age peak at 45 to 50 years. Besides the aiding tool and the age there are no other features in the data set that can easily distinct these two groups. The users of Valys that do not travel partly by train but possibly have the incentive to do so split similar into two personas, but now the users with aiding tool are divided over a bigger age range. The users with aiding tool also book on average three more days ahead than users without aiding tool.

One can predict whether a Valys user travels partly by train based on the distance of the trip, the amount of budget left at the end of last year, the amount of trips made partly by train last year and the age of the user. Based on these criteria 84% of the trips can be classified correctly. The used aiding tool seemed to have some influence in whether someone would make a transfer or not by train. There were no other features that gave a clear distinction in routing. It is possible that preferences in routing are more of a personal matter, not completely deducible from the known characteristics of the user. Another possibility is that currently not enough data is stored about the user and the trip.

Currently Valys users that travel partly by train are a small group compared to the users that do not travel partly by train. The group with users that travel partly by train can be increased, but only by users who have an incentive to do so. The distances of the trips made by the user need to be large enough that travelling partly by train is possible and the user needs to already spend most of his or her budget, since this could indicate he or she would like to make more trips. The age is also taken into account, older users have a lower change of starting to travel partly by train. The current maximum increase in number of trips partly by train is estimated at 9.6 % of the total number of trips. Even after this selection it is still possible that some users will not travel partly by train. Reasons for this include they are really not able to (medical information is not available), the budget is exactly enough, they are not interested in making more trips with Valys or they are just to stubborn in not wanting to travel partly by train. More than half of the trips

(57.6%) will probably never be done partly by train.

Valys users seem to want a more personalized journey planner. The following criteria are found to be important by the users: travel time, number of transfers, frequency, walking distance, budget and cost. Mentioned restrictions that could be taken into account are maximum walking distance, walking speed and maximum number of transfers. Currently a lot of time is reserved for walking and waiting when travelling partly by train and this causes the travel time to be much higher than when not travelling partly by train. When the system is made more personalized and the system is in use for a while this could give better knowledge about the routing preferences of the Valys user.

II

JOURNEY PLANNER WITH PREFERENCES

6

INTRODUCTION

Having mobility restrictions is a big problem for mainly the older target group in the Netherlands. In 2014 10.3% of the male population and 24.7% of the female population in the Netherlands aged 65-74 years had a problem with their mobility [8]. For the age group 75 years and older this is even higher, 29.9% of the male population against 48.8% of the female population had a problem with their mobility. People with mobility restrictions are not always able to visit for example family and friends on their own.

Valys facilitates social and recreational transport for people with mobility restrictions who travel more than five public transportation zones. This also includes younger people who use a wheelchair or mobility scooter. A Valys trip can be completely by taxi or partly by train and taxi. The user group of Valys consists of a broad range of users who have different routing preferences for trips partly by train. The current journey planner of Valys can not take user preferences into account.

The outline of this chapter is as follows. Section 6.1 explains how Valys works in more detail. Next section 6.2 formulates the problem that needs to be solved in a more general way that can apply to more applications than just Valys. At last section 6.3 gives an outline of this part II.

6.1. VALYS

In 2016 almost 970000 trips were made by Valys. Valys facilitates three different travel options: Valys Basis, Valys Begeleid and Valys Vrij. Valys Basis transports Valys users completely by taxi and these taxi rides are combined with other Valys users. Both Valys Begeleid and Valys Vrij trips are partly by train. A common trip partly by train first brings the user by taxi to a train station, the user takes the train to another train station and from there the user takes a taxi to his or her destination. The difference between a Valys Begeleid and Valys Vrij trip is that a Valys Begeleid trip offers assistance at the train station and this is done at 31 train stations in the Netherlands.

At the beginning of the year the *Persoonlijk Kilometer Budget* (PKB) of the Valys user is set again to its initial value. For most users this is 600 km and for users with a high

PKB this is 2250 km. A valys user can qualify to get a high PKB budget when he or she cannot travel by train due to a medical condition or other restriction. The user only loses kilometers from his or her budget when using the taxi. When the user completely uses his or her budget, travelling by taxi becomes more expensive, the price per kilometer increases from €0.20 to €1.31. This means more trips for the same amount of money are possible when travelling partly by train.

The current journey planner of Valys has a couple limitations. The first limitation is that it only takes into account the three nearest train stations to the location of departure for the taxi to drive to and the three nearest train stations to the location of arrival for the taxi to drive from. Next it only calculates the fastest route between these train stations, other criteria such as number of transfers, frequency, walking distance, budget and cost are not taken into account. Furthermore the user cannot indicate restrictions for the trip, such as the maximum walking distance. Another limitation is that the transfer time is not based on the actual walking distance or walking speed of the user. The last limitation is that the user always departs exactly at the departure time given by the user.

6.2. RESEARCH

This part II investigates how to make a better and more personalized journey planner that takes into account observations from part I. For Valys it is important that the order of transportation modes is first taxi, then one or multiple trains and next one taxi. The goal of the journey planner is to attract more Valys users to travel partly by train and to improve the journey for Valys users that already travel partly by train. The journey planner should be fast enough and remove the limitations defined in the previous section.

When defining the problem more generally the interest lies in creating a personalized journey planner that can take into account user preferences and multiple modes of transportation. Preferences include restrictions on the route and criteria to optimize on. Mainly for large distances there exist a lot of possible routes from the departure to the arrival location. Only a limit amount of routes can be shown to the user.

6.3. OUTLINE

The outline of this part II is as follows. Chapter 7 gives an overview of related work. Next chapter 8 formally formulates the problem that needs to be solved. Chapter 9 introduces a solution that can compute all Pareto-optimal routes from source to target and chapter 10 shows how this can apply to Valys. Next chapter 11 discusses and compares three methods that can select a subset of Pareto-optimal routes. Chapter 12 compares the current Valys journey planner with the journey planner introduced in this thesis. Finally the contributions of this thesis are summarized in chapter 13.

7

RELATED WORK

The most famous algorithm to solve the shortest path problem is Dijkstra's algorithm. Since the publication of this algorithm in 1959 by E. W. Dijkstra [9], many improvements of this algorithm have occurred and other methods to solve the shortest path problem are introduced. The networks the shortest path problem applies to can be split up into two categories: road networks and public transportation networks. Road networks apply to unrestricted transportation modes and public transportation networks apply to schedule-based transportation modes. Each network has its own suitable algorithms to solve the shortest path problem and these algorithm are discussed in section 7.1 and 7.2. Section 7.3 explains the concept of Pareto optimality and how this can apply to route planning.

7.1. ROAD NETWORKS

The section discusses current approaches to solve the shortest path problem in a road network. Basic Dijkstra is one of them, but works too slow in large networks. Road networks apply to different modes of transportation; e.g. car, bike and walking.

Another approach to solve this is with Contraction Hierarchies. The concept of Contraction Hierarchies in road networks was first introduced by R. Gesiberger et al. [10] and is based on the concept of node contraction. In the precomputation step the nodes in the graph are contracted in order of (ascending) importance. The Contraction Hierarchy (CH) is split into two graphs: the upward graph and the downward graph. The query performs a bidirectional Dijkstra shortest-path search. R. Gesiberger et al. [10] claim that CHs are well suited for many-to-many routing. In 2013 S.Funke et al. [11] refined and enhanced the precomputation step such that CHs can work for multi-criteria objectives. A lesser known approach is based on Graph Separators and is for road networks first introduced by D. Delling et al. [12]. This approach is slower than Contraction Hierarchies, but is still suitable for dynamic scenarios.

7.2. PUBLIC TRANSPORTATION NETWORKS

This section discusses current solutions to solve the multi-criteria shortest path problem in a public transportation network. Public transportation networks apply to different modes of transportation; e.g. train, bus, metro and tram. These modes of transportation are based on a schedule.

7.2.1. DIJKSTRA

The traditional approach of solving the shortest path problem in a public transportation network is based on Dijkstra. Y. Disser et al. [13] for example present a prototype to solve the problem of finding all Pareto-optimal solutions in a multi-criteria setting of the shortest path problem in time-dependent graphs. Their solution is based on a multi-criteria generalization of Dijkstra. Multi-dimensional labels are used to remember all promising paths with which a node was reached. These labels are associated with the nodes. For large networks, Dijkstra's algorithm is too slow to compute the shortest path.

7.2.2. RAPTOR

Not every solution is Dijkstra based. One of these solutions is called RAPTOR and is introduced by D. Delling et al. [14]. RAPTOR uses dynamic programming to operate directly on the timetable, works in rounds and is a magnitude faster than accelerated Dijkstra-based approaches when optimizing on travel time and number of transfers.

In round k it computes the fastest way of getting to every stop with at most k trips (which is k-1 transfers). For every stop the earliest arrival time for at most 1, 2,... and k trips is stored. Initially the earliest arrival time is set to infinity for every stop except the source stop, which is set to the departure time of the query.

The stops for which the arrival time improves during a round are marked and these stops are used as input for the next round. In the first round the only marked stop is the source and k is 1. Every round RAPTOR iterates only through the routes that contain a marked stop. When scanning a route r, every stop of r is iterated over to keep track of the earliest possible trip of r that can be taken. If this current trip achieves an earlier arrival at a stop than the current best arrival at that stop, the current best arrival time is updated. Also the earliest arrival time for the stop with at most k trips is updated. When no stops are marked the algorithm is done. RAPTOR scans a route at most once per round.

Finding the Pareto-optimal solutions for more criteria than travel time and number of transfers is done with the use of bags. These bags store non-dominating labels for each stop p in round k. Multi-criteria RAPTOR is called McRAPTOR. As RAPTOR does not rely on preprocessing (of complete routes) it can be easily used in dynamic scenarios.

7.2.3. CSA

Another not Dijkstra based approach to find the optimal route(s) in a transportation network is called the Connection Scan Algorithm (CSA) and is introduced by J. Dibbelt et al. [15]. The core of this algorithm is based on a list of connections ordered by departure time. CSA requires only to iterate once through this list and this results in very fast query times. While iterating through this list the algorithm keeps track for every stop which connection results in the earliest possible arrival at that stop. A connection in this con-

text is a connection between two stops without an immediate stop in between which has a departure time and an arrival time. Just like RAPTOR, CSA does not use the time expensive priority queue of Dijkstra and does not rely on preprocessing (of complete routes). It can easily adjust to changes in the timetable and this makes it easily usable in dynamic scenarios.

J. Dibbelt et al. [15] extend CSA to compute multi-criteria profiles that optimize on departure time, arrival time and number of transfers. For every stop a partial profile is stored. When optimizing on departure time and number of transfers, Pareto-optimal pairs of (departure time, arrival time) are stored. When also optimizing on the number of transfers the profile pairs become more complex. Profiles are stored as (arrival time, bag) pairs, where a bag is a (departure time, number of transfers) pair. This extended CSA algorithm (pCSA-CT) computes the union of bags and removes dominated entries. It is not demonstrated how the profile pairs look like for more than three criteria. J. Dibbelt et al. [15] indicate there is a future work possibility for combining CSA with existing techniques developed for road networks.

7.2.4. TRANSFER PATTERNS

Another algorithm to find the optimal route in a public transportation network is called Transfer Patterns, this algorithm addresses challenges faced by Google Maps and is presented by H. Bast et al. [16]. It uses the fact that many shortest paths share the same transfer patterns: a sequence of stations where a change of vehicle occurs. It precomputes a part of these transfer patterns, so that the optimal transfer patterns can be computed from this.

The data structure for direct-connection queries are precomputed as follows. A sequence of stops without needing a transfer is called a trip. Trips with the same sequence of stops are stored in a line, an example line is displayed in table 7.1. The stations and trips are sorted by departure time. For each stop the corresponding lines and its position are computed (this is called the incidence list). Example incidence lists are:

stop 1: (line a,2) **(line x,1)** (line y,3) ...
stop 3: **(line x,3)** (line b,4) (line z,4) ...

To answer a direct-connection query between stop 1 and 3, the incidence lists for the two stops is intersected on line number and for the example given above this is x. When the departure time of the query is 09:05, one could depart 09:15 from stop 1 and arrive 09:45 at stop 3.

Table 7.1: Transfer pattern example line.

line x	Stop 1	Stop 2	Stop 3	...
trip 1	09:00	09:15 09:16	09:30 09:31	...
trip 2	09:15	09:30 09:31	09:45 09:46	...

From the precomputed data all combinations that produce a transfer pattern between every two stops are computed. Next these patterns are overlayed and this is called the query graph. Each edge in the query graph is a direct connection without any transfers in between. To find the optimal connections between the source and the target a

shortest path search is performed on the query graph. Although the query times of this algorithm are very fast, the precomputation of the transfer patterns can take hours for large graphs and this is just for optimizing on travel time and number of transfers.

7.2.5. COMPARING ALGORITHMS

H. Bast et al. [17] compare these four solutions for solving the multi-criteria shortest path problem on runtime. Dijkstra is the slowest. CSA is by a factor of two faster than rRaptor (profile extension of Raptor), both algorithms were optimizing on travel time and number of transfers when the running time was measured. This leaves CSA and transfer patterns. Transfer Patterns is faster than CSA, but due to the time needed for preprocessing of complete routes (especially for many criteria), one could prefer CSA over transfer patterns when wanting to optimize on many criteria. CSA is also conceptually simpler.

7.3. PARETO-OPTIMAL ROUTES

When taking multiple criteria into account, there can exist multiple interesting routes from source to target. A common approach too decide which routes to keep is based on Pareto-optimality. Route r_1 Pareto dominates route r_2 when it is better than r_2 in at least one criteria and no worse than all other criteria. Calculating all Pareto-optimal routes from source to target can lead to a large number of routes.

H. Bast et al. [18] propose an approach to compute a small representative set of reasonable routes from a Pareto-optimal set of routes in a multi-modal scenario. It takes three criteria into account: velocity, availability and costs. The approach is based on removing unreasonable routes. It removes routes that are unreasonable based on the combination of transportation modes. For example a route where only the car is used for a long time is reasonable, but a route where both transit, walking and car is used each for a long time is not reasonable. This approach removes the unreasonable routes, but does not take into account that the number of reasonable routes can still be (too) large to show to a user.

D. Delling et al. [19] propose to score the routes from the Pareto set in a postprocessing step using techniques from fuzzy logic and to only show the routes with the k highest score to the user. M. Farina et al. [20] define fuzzy logic for many-criteria optimization problems based on the limitations of Pareto optimality when more than two or three objectives are taken into account. When comparing two solutions the following aspects are not taken into account: the number of improved/decreased objectives, the size of such improvements/decreasements and preferences between objectives (if any).

A different Gaussian function is used for every objective to indicate how much better or worse one objective of a journey is compared that objective for another journey. This Gaussian function $\mu_=(x) := exp(\frac{\ln(\chi)}{\epsilon^2}x^2)$ indicates the equality operator between two different values of the same objective, where $0 < \chi < 1$ and $\epsilon > 0$ are different for every objective and x indicates the difference between two values for the same objective. Besides the equality operator two different operators exist as is described by T. Pajor [21]:

- $\mu_<$:= 1-$\mu_=(x)$ if x < 0 else 0
- $\mu_>$:= 1-$\mu_=(x)$ if x > 0 else 0

T. Pajor [21] denotes $n_b(J_1, J_2)$ as the fuzzy number of criteria in which journey J_1 is better than journey J_2, this is equal to the summation of $\mu_<$ for every objective between journey J_1 and journey J_2. Next $n_w(J_1, J_2)$ is denoted as the fuzzy number of criteria in which journey J_1 is worse than journey J_2, this is equal to the summation of $\mu_>$ for every objective. These two fuzzy numbers can be used to define the degree of domination $d(J_1, J_2)$ (how much J_1 dominates J_2):

$$d(J_1, J_2) = \begin{cases} 0 & \text{if } n_b(J_1, J_2) \leq n_w(J_1, J_2) \\ \frac{n_b(J_1, J_2) - n_w(J_1, J_2)}{n_b(J_1, J_2)} & \text{otherwise} \end{cases} \quad [21]$$

In this equation, $d(J_1, J_2) = 0$ means that J_1 does not dominate J_2, otherwise J_1 fuzzy-dominates J_2 by degree $d(J_1, J_2)$. This degree of domination can be used to calculate a score for every route. D. Delling et al. [19] take four criteria into account: arrival time, walking time, transfers and cost.

8

PROBLEM DEFINITION

This chapter explains how the current journey planner of Valys works in section 8.1 and enumerates its limitations in section 8.2. Next section 8.3 formulates the problem, that needs to be solved to remove these limitations, in such a way that it can apply to more applications than just Valys. Furthermore section 8.4 discusses current literature. Finally the research questions are formulated in section 8.5 and the scope is defined in section 8.6.

8.1. CURRENT JOURNEY PLANNER

The current journey planner of Valys gives the user just as any other journey planner the possibility to enter the desired departure location, arrival location, date of departure and time of departure. Additionally the user can state the aiding he or she will bring if applicable. The journey planner returns seven routes and indicates for each route the cost, the number of needed taxi kilometers and the expected travel time. One route is completely by taxi and the other six routes are partly by train. Every route starts exactly at the departure time given by the user. A common trip partly by train consists of a taxi to a train station, a train to another train station and a taxi from that train station to the destination.

The routes partly by train for a Valys Vrij trip are computed as follows. First the distance and time from the arrival location time to the three nearest train stations by taxi are computed. Next the possible routes from these train stations to the three nearest train stations of the arrival location by train are computed. At last the time and distance by taxi to the arrival station from these train stations are computed. In total this results in nine different routes. Of these routes the fastest, the cheapest and the best (based on travel time and cost) is shown to the user. For most train stations the journey planner takes a transfer time of 15 minutes from and to the taxi and 10 minutes between train transfers into account.

The difference between Valys Begeleid and Valys Vrij is that Valys Begeleid offers assistance at train stations, which is done for 31 train stations in the Netherlands. To com-

pute the routes for a Valys Begeleid trip the only difference is that users can only depart, arrive and transfer at Valys Begeleid stations.

8.2. CURRENT LIMITATIONS

As the current journey planner takes a limited amount of train stations into account to drive to or from by taxi, faster or in another way better routes may not be found. This also causes the fastest, cheapest and best route to be most of the time the same route. The best route between two train stations is only based on travel time, other criteria such as number of transfers, walking distance, budget and frequency are not taken into account.

The user cannot indicate restrictions for the trip. Some users cannot walk more than x meters or will never want to take more than y train transfers, but still these routes are shown to the user. The transfer time that the system takes into account is for some users more than needed, it is not based on the walking distance or walking speed of the user. The transfer time is the same for all transfers on a train station, even though for some transfers walking to the other side of the train station is required and for others transfers only walking to the other side of the platform is required.

Computing each of the nine possible routes is done separately and this causes in a worst case scenario that the routes appear three seconds after the user gives input (departure location, arrival location etc.). The last limitation of the journey planner is that the user always departs exactly at the departure time indicated by the user, even though there might exist a route that departs earlier or later with a lower travel time.

8.3. FORMAL PROBLEM

This section translates the limitations from the previous section into a more generalized problem. The problem is about a user who gives as input a departure location, arrival location, departure time, date of departure and possibly some restrictions for the route. The output should be given almost instantly and should consist of a small representative list of routes from the departure location (source) to the arrival location (target).

Planning a route from source to target can be described with the shortest path problem. In graph theory the shortest path is described as the edges that form together a path from the source vertex to the target vertex while the sum of the weight of these edges is minimized. In route planning this weight is most of the time the distance, but could be many things.

The route should combine multiple transportation modes and this is called multi-modal. Two types of transportation modes are taken into account: unrestricted and schedule-based transportation modes. Unrestricted transportation modes such as walking, biking and driving are not based on a schedule and can departure at any moment. On the other hand schedule-based transportation modes such as bus, train, metro and tram do run on a schedule. The problem focuses on first one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode. It also assumes a constant travel time for the unrestricted transportation mode. When wanting to take into account multiple modes of transportation and to optimize on multiple criteria the problem specializes into the multi-modal multi-criteria shortest path problem.

8.4. CURRENT LITERATURE

Solving the multi-criteria shortest path problem in public transportation networks is in current literature focused on a limited number of criteria (at most 3). The most common criteria that are optimized on are travel time, number of transfers and cost. CSA is a conceptual easy algorithm and is the fastest multi-criteria shortest path algorithm for public transportation networks that does not need to preprocess complete routes. There also exist a research possibility to combine CSA with existing techniques developed for road networks. Currently fuzzy dominance is the only method that can make a selection from a Pareto-optimal set of routes, which is focused on more than two or three criteria and does more than filtering out unreasonable routes.

8.5. RESEARCH QUESTIONS

Summarizing the problem described in the previous sections leads to the following research question:
How can a representative set of routes be found that is not too big from source to target that combines multiple transportation modes, takes into account many criteria, can take into account restrictions, is fast enough and focuses first on one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode?

- How can CSA be combined with existing techniques developed for road networks?
- How can multi-criteria CSA be extended to work for many criteria (more than three)?
- What is the influence on the travel time when the Valys user is able to depart earlier or later?
- What is the influence on the travel time when a restriction is put on a criterion?
- How can a subset of routes be chosen from a Pareto-optimal set of routes when the Pareto-optimal set of routes is too large? Is fuzzy dominance a suitable method for this in the context of Valys?
- How much improvement can be achieved for the Valys user in comparison to the current journey planner when looking at travel time, number of transfer, budget and cost?

8.6. SCOPE

This section defines the scope of the research. As said before this research is limited to first one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode. When computing the distance and time by the unrestricted transportation mode, real-time traffic information is not taken into account. Also no real-time information about the public transport is taken into account, only static transport data is used. When calculating the walking distance, the fastest route for someone who is not in a wheelchair is chosen.

Methods created to answer the research questions described in the previous section are tested on Valys trips partly by train from last year (made between 01-01-2016 and 31-12-2016) which are delivered by Transvision [6]. These trips consists of Valys Begeleid and Valys Vrij trips, although all trips are considered as Valys Vrij trips since more improvement is possible for this Valys type. Of these trips only the postal code of the departure location and the postal code of the arrival location are used. These postal codes are converted to latitude and longitude with the Google Maps Geocoding API [22].

9

PARETO-OPTIMAL ROUTES

This chapter describes a solution to find all Pareto-optimal routes from source to target for many criteria and multiple modes of transportation. The solution focuses on first one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode. The outline of this chapter is as follows. Section 9.1 describes how CSA can be used to compute the route with earliest arrival for the indicated order of transportation modes and section 9.2 how this be adapted to work for multiple criteria. Next section 9.3 shows how this algorithm can be accelerated. At last section 9.4 shows how the algorithm is adapted to be able to depart earlier or later than the indicated departure time of the user.

9.1. EARLIEST ARRIVAL

J. Dibbelt et al. [15] indicate the existence of a future work possibility for combining CSA with existing techniques developed for road networks. Currently CSA only takes into account schedule-based transportation modes. This section describes how CSA can be used to compute the route with earliest possible arrival for first one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode. Besides earliest arrival, no other criteria are taken into account in this section to make it easier to explain. Figure 9.1 gives a schematic representation of the journey.

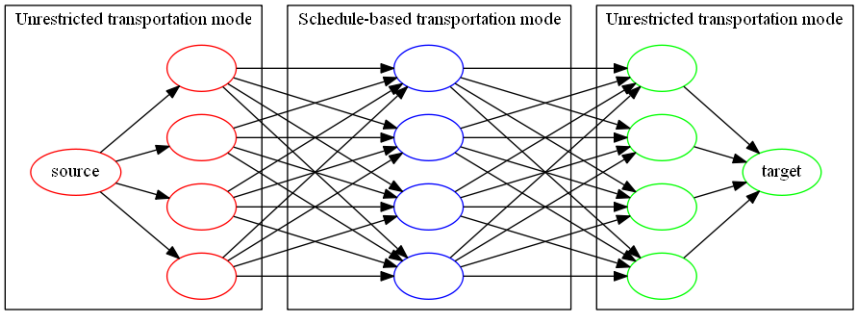


Figure 9.1: Journey from source to target. First one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation. This figure is made with Graphviz [3]

.

To compute the earliest arrival, the shortest path based on arrival time is needed for a combination of the road network and the transportation network. It is possible to combine these two networks into one, but that results in a really large network and could result in a shortest path that does need meet the given order of transportation modes (such as an unrestricted transportation mode between two schedule based transportation modes). Another possibility is to convert one network into the other and solve the shortest path problem with the algorithm best for that network. As follows from chapter 7 the most suitable algorithm to solve the shortest path problem (when taking multiple criteria into account) for a transportation network is CSA and for a road network is Contraction Hierarchies. Given that CSA is conceptually simpler and that only the roads between the source and potential stops and the roads between the potential stops and the target are needed it seemed best to convert the road network into a transportation network.

The first step is to compute the time needed to get from the source to potential stops in a road network with an unrestricted transportation mode and the time needed to get from potential stops to the target with an unrestricted transportation. This is equal to solving the one-to-many and the many-to-one shortest path problem in a road network and this can be solved with Contraction Hierarchies. Our solution does not go into more detail as the focus of this thesis is more on combining multiple modes of transportation and being able to optimize on many criteria.

The Connection Scan Algorithm (CSA) is based on the concept of connections. The schedule-based transportation modes can easily be converted to connections as these modes have a fixed arrival and departure time for every stop. An unrestricted transportation mode does not have a fixed departure and arrival time for every stop and so this needs to be determined. As the first mode of transport from the source is an unrestricted transportation mode and only optimization on arrival time takes place, the departure time of this transportation mode can be set to the departure time given by the user. The arrival time can be set to this departure time plus the time needed for the unrestricted transportation mode to get to a potential stop. For every potential stop, a

connection is created that departs from the source and arrives at that potential stop.

After CSA finishes walking through the list of connections that consist of schedule-based connections and connections from source to potential stops only the fastest connection to the target is not yet determined. This is determined by considering all potential stops to the target as connections to the target, the connection that makes the earliest arrival possible at the target is the final connection of the journey.

The pseudocode of multi-modal CSA is displayed in algorithm 1 and 2. The input parameters are the date, departureTime, sourceStops and targetStops. The sourceStops are the potential stops from the source and the targetStops are the potential stops to get to the target. The output of multi-modal CSA is the journey. The bestConnections list keeps track for every stop (including the source and target) which connection causes the earliest arrival at that stop. A connection is updated if a new connection is found that makes an earlier arrival possible than the current connection and which departs later than or equal to the arrival time of the previous connection. For this to work the best connection to the source stop needs to be set to a connection with the arrival equal to the departure time given by the user.

Algorithm 1 Multi-modal CSA

Input: date, departureTime, sourceStops, targetStops
Output: journey

Read in timetable
timetable ← read connections from file belonging to the date

Initialization
numStops ← total number of stops
source ← numStops + 1
target ← numStops + 2

bestConnections ← ∅
for i = 1 to numStops **do**
    add empty Connection to bestConnections
end for
c ← Connection to the source that arrives at departureTime
add c to bestConnections
add empty Connection to bestConnections

First unrestricted transportation mode
for Stop s in sourceStops **do**
    c ← Connection from source to s (which leaves at the departureTime and arrives at the departureTime + travel time to s from source)
    add c to timetable
end for

CSA()

Second unrestricted transportation mode
for Stop s in targetStops **do**
    if current Stop leads to an earlier arrival at the target **then**
        update current Connection to the target in bestConnections
    end if
end for

Print journey from bestConnections

Algorithm 2 CSA

```
1: sort timetable on departure time
2: for Connection c in timetable do
3:     if c departs later or at the same time as the previous Connection &
       c arrives earlier than the current Connection then
4:         update current Connection to c in bestConnections
5:     end if
6: end for
```

9.2. MULTI-CRITERIA

J. Dibbelt et al. [15] describe how CSA can be extended to compute multi-criteria profiles, where triples of departure time, arrival time and number of transfers are optimized. They maintain profiles as (arrival time, bag) pairs, where a bag is a (departure time, number of transfers) pair. Whether it works for more criteria and if it is fast enough is not described.

The previous section described how CSA can be used to compute the route with earliest arrival for first one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode. This section explains how the algorithm from the previous section can be adapted such that all Pareto-optimal routes from source to target can be computed for many criteria.

At first, the implementation of multi-criteria profiles for CSA by J. Dibbelt et al. [15] seemed to be an approach that would not work fast enough for more criteria. Fast enough in this context means that when the user gives input, shortly afterwards the possible journeys should appear to the user. After that the idea arose to not only calculate the fastest connection per stop as in the previous section but to calculate the fastest connection per stop under any possible combination of restrictions and these restrictions are based on the criteria that need to be optimized on. For example, if besides arrival time optimization on number of transfers and walking distance is required, the fastest connection must be calculated per stop for each possible combination of these two criteria as maximum:

- Number of tranfers: 0, 1, 2 etc.
- Maximum walking distance: 0 m, 1 m, 2 m, 3 m etc.

It is not necessary to do this for an infinite number of transfers and walking distances, because in every context there is a maximum acceptable value for each criterion. However, it appeared that when multi-criteria CSA is computed in this way, some Pareto-optimal routes are not calculated. Therefore, it has been re-examined to implement multi-criteria in a similar way to the multi-criteria implementation of J. Dibbelt et al. [15].

Instead of only storing the fastest connection per stop, now every stop stores a list of non-dominated tuples. A tuple consists of the current values for the criteria that are of interest, the last connection and a reference to the previous tuple. When a new connection c is considered that departs from A and arrives at B, the list of tuples of A and B are

requested. For every tuple t in the list of A, it is first determined whether the connection c departs after the connection of tuple t arrives. If so the values for the new tuple, of adding connection c after the connection of tuple t, need to be computed. For this new tuple, the previous tuple is t and the last connection is connection c. Another requirement is that this new tuple has values for the criteria below the maximal acceptable values and these maximum acceptable values can be set by the user and/or determined in the given context. The pseudocode of this is displayed in algorithm 3.

If this new tuple dominates a tuple in B, this tuple in B is removed. If the new tuple is not dominated by one of the tuples in B, the new tuple is added to the tuple list of B. The algorithm that determines the domination is displayed in algorithm 4.

Tuple 1 dominates tuple 2 (based on minimization) when:

- Every criteria has a lower or equal value for tuple 1 than for tuple 2
- At least one criteria has a lower value for tuple 1 than for tuple 2

The first unrestricted transportation mode is added as connections to the timetable just as is displayed in algorithm 1. Algorithm 5 displays how the second unrestricted transportation mode is added to the equation.

Algorithm 3 Pareto optimal CSA

Input: timetable, numStops, source, departureTime
Output: bestTuples

```
1:  bestTuples ← ∅
2:  for i = 1 to numStops do
3:      add empty Tuple list to bestTuples
4:  end for
5:  cc ← Connection to the source that arrives at departureTime
6:  departureTuple ← Tuple containing cc, initial criteria values and empty Tuple
7:  add departureTuple to bestTuples
8:  add empty Tuple list to bestTuples

9:  sort timetable on departureTime
10: for Connection c in timetable do
11:     departureTuples ← get the Tuples of the departure location of c
12:     arrivalTuples ← get Tuples of the arrival location of c
13:     for Tuple t in departureTuples do
14:         if currentArrival of t ≤ departure time of c then
15:             newTuple ← possibleTuple(t,c)
16:             if tupleValuesBelow(newTuple) then
17:                 determineDomination(newTuple,arrivalTuples)
18:             end if
19:         end if
20:     end for
21: end for
```

Algorithm 4 determineDomination

Input: newTuple, arrivalTuples

```
1: removeList ← ∅
2: dominated ← false
3: for Tuple t in arrivalTuples do
4:     if newTuple dominates t then
5:         add t to removeList
6:     end if
7:     if t dominates newTuple then
8:         dominated = true
9:     end if
10: end for
11: remove tuples in removeList from arrivalTuples
12: if !dominated then
13:     add newTuple to arrivalTuples
14: end if
```

Algorithm 5 Second unrestricted transportation mode

Input: bestTuples, targetStops, target

```
1: for Stop s in targetStops do
2:     departureTuples ← get Tuples of s
3:     for Tuple t in departureTuples do
4:         Tuple newTuple = possibleTuple(t,s,target)
5:         if tupleValuesBelow(newTuple) then
6:             determineDomination(newTuple,Tuples of the target)
7:         end if
8:     end for
9: end for
```

9.3. CSA ACCELERATED

Not all connections need to be walked through by CSA. After sorting the connections on departure time a binary search can be performed to find the first connection with a departure time equal to the departure time of the user, the index of this connection is where CSA starts. Another binary search can be performed to find the last connection with a departure time x hours later than the departure time of the user, the index of this connection is where CSA stops. The assumption is that routes that start earlier than the departure time or routes that are larger than x hours are not of interest to the user. This last binary search can only be performed if x is given as input by the user or if there is another way x can be determined.

9.4. DEPARTING EARLIER OR LATER

So far the algorithm assumes the user always departs exactly at the departure time given by the user, even though a shorter travel time is possible if one would departure earlier or later.

When only optimizing on arrival time it is possible to perform CSA and based on the found earliest arrival perform CSA backwards to determine whether a later departure time is possible with the same arrival time. However for more criteria this would work differently. When CSA is calculated backwards for multiple criteria, some connections may change and this could also change the values for the criteria and possibly even its Pareto-optimal status. Therefore departing earlier or later is tackled in a different way.

Departing earlier or later is solved by adding more unrestricted connections from source to potential stops. Figure 9.2 shows the needed connections when allowing for example to depart 10 minutes earlier to 10 minutes later for one potential stop. The total travel time replaces earliest arrival as criteria to optimize on.

Another adaption is on the number of tuples that are initially added to the source. Before only one tuple is added to the source as is described in lines 5 to 7 in algorithm 3. To allow the user to depart earlier or later than the given departure time, multiple tuples with different departure times need to be added as is displayed in algorithm 6, which shows an adaption of algorithm 3. This algorithm allows the user to depart 10 minutes earlier to 10 minutes later.

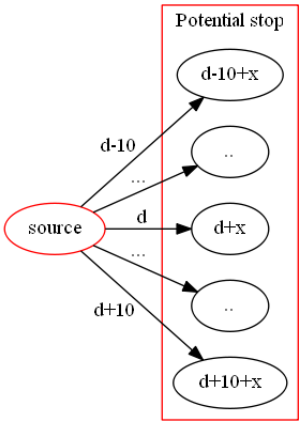


Figure 9.2: Needed connections for departing 10 minutes earlier to 10 minutes later from start to a potential stop, where d is the departure time given by the user and x the travel time from start to the potential stop. This figure is made with Graphviz [3]

Algorithm 6 Pareto optimal CSA (adaption of algorithm 3)

```
    ...
5:  for x = -10 to 10 do
6:      cc ← Connection to the source that arrives at departureTime + x
7:      add departureTuple to bestTuples
8:  end for
    ...
```

10

PARETO-OPTIMAL VALYS ROUTES

This chapter describes how the solution from the previous chapter can apply to Valys. Three experiments are performed on data of Valys trips from last year (2016). The first experiment computes the number of Pareto-optimal routes from source to target to determine if this number of routes is sufficient or too much to show a user. The second experiment researches the gain in travel time when being able to depart earlier or later. The third experiment shows what the influence is on the travel time when a restriction is put on a criterion. This is done to determine how much extra travel time is needed for certain restrictions and to determine for which distances in combination with which restriction no route exists.

10.1. PARETO-OPTIMAL ROUTES

The solution in the previous chapter described how all Pareto-optimal routes from source to target can be found for many criteria and multiple modes of transportation. The solution focuses on first one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode. In the case of Valys the unrestricted transportation mode is the taxi and the schedule-based transportation modes are trains. In the future trams, buses and metros might be added as schedule-based transportation modes to Valys, but currently the focus is on just taxi-train-taxi trips in the Netherlands.

Besides the travel time five other criteria can be of interest to the Valys user, as is argued in part 1 of this thesis and are therefore optimized on. These criteria include:

- Number of transfers
- Frequency
- Walking distance
- Budget

- Cost

All these criteria, except for the frequency, need to be minimized. The frequency is defined per connection c and is equal to the number of connections that have the same departure stop and arrival stop as connection c which depart within an hour of connection c. The budget is the number of kilometers by taxi. After a trip is booked this number is reduced from the PKB.

Users may require restrictions on some of the criteria. Restrictions include:

- Maximum number of transfers
- Minimum frequency
- Maximum walking distance
- Maximum budget
- Maximum additional travel time compared to the trip completely by taxi
- Maximum additional cost compared to the trip completely by taxi

Now only Pareto-optimal routes that are within the desired restrictions are stored. Additionally it is possible to adjust the walking speed, this effects the transfer time needed for transfers from taxi to train, train to train and train to taxi. Section 10.1.1 describes which part of the process can be done before the user plans his or her journey and section 10.1.2 explains what part of the process can only be done after the user gives his or her user input (departure location, arrival location, departure time etc.).

10.1.1. PREPROCESSING

This subsection explains which part of the process can be done before the user plans his or her journey. Since the departure and arrival location are unknown, nothing about the taxi part can be precalculated.

First of all the CSA algorithm needs a list of connections, one for each day to iterate over, beforehand this list can be generated from GTFS (General Transit Feed Specification) data [23] which is a common format for public transportation schedules. After the list of connections is created, the frequency can be calculated and added for every connection. Next the train cost between every two train stations is precalculated with data from *Rijden de treinen weblog* [24]. The train cost are only based on NS rates, other transport companies are not taken into account.

At last all possible walking distances per station are precalculated, this includes walking between every two train platforms and walking between the taxi stand and every train platform. The walking distance is calculated with OpenTripPlanner (OTP) on walk only mode [25]. The required locations of the taxi stand are delivered by Transvision [6], but are not yet accurate enough. The location of the train platforms (in latitude and longitude) is included in the GTFS data [23]. In most cases OTP returns an accurate walking distance, sometimes not. When a walking distance higher than 500 m is returned, it is assumed something went wrong and this walking distance is set to be 500 m. When the walking distances can be calculated more accurately the overall program will also work

better. The walking distance combined with the walking speed determines the needed transfer time. Table 10.1 displays the runtime for each preprocessing step. Precalculating the walking distances requires the most time, fortunately this step does not need to be done daily. The high runtime is a result of the many one-to-one requests performed by OTP to calculate the walking distances, when this would be changed to many-to-many requests the runtime would be a lot lower. Every station would need one many-to-many request.

Table 10.1: Runtime per preprocessing step

Preprocessing step	runtime
Connections for one day	16 s
Frequency for one day	1 s
Train cost	280 ms
Walking distances	23 min

10.1.2. AFTER USER INPUT

After the user gives his or her departure location, arrival location, time and date of departure the Pareto-optimal routes from the departure location to the arrival location are computed.

The first step is to find the potential train stations to drive to from the source and the potential train stations to drive from to the target. The potential train stations for the source are the train stations that are within a distance from the source that is lower than the distance from the source to the target. The potential stops for the target are the stops that are within a distance to the target that is lower than the distance from source to the target. This is visualized in figure 10.1.

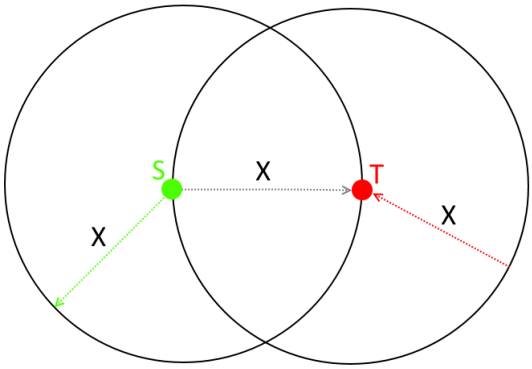


Figure 10.1: The distance between source (S) and target (T) is equal to x. To potential stops for the source are within a distance of x from the source and the potential stops for the target are within a distance of x to the target.

The potential stops for the source and target are selected based on as the crow flies

distance from latitude and longitude coordinates. A tool called the Travel Matrix of Calendar42 [26] is used to calculate the actual distance and time by car from the source to potential stops and from potential stops to the target. This tool uses Contraction Hierarchies to perform one one-to-many and one many-to-one request. The average runtime for performing the two request simultaneously is about 78 ms (with a standard deviation of 36 ms).

The decision to not implement Contraction Hierarchies, but use a tool that can perform this is made for different reasons. When wanting to optimize on the six criteria as described in section 10.1 only the travel time and the travel distance need to be computed with CHs. The criteria budget and cost can be deduced from the distance. The criteria frequency and number of transfers do not need to be computed, because the taxis can leave at any time and there will be no transferring from one taxi to another (directly). Calculating the walking distance from the taxi to the train platform needs to be done separately as walking requires a different shortest path than for a car. When only the distance and time need to be known of the best route by car, there will be no addition to the current literature, unless the complete devotion of the thesis is to Contraction Hierarchies. Finally CHs is not something that can be implemented in a short amount of time.

Now that the possible taxi rides are known, CSA can start iterating over the connections as is described in the previous chapter. Every tuple stores the latest connection, the previous tuple and the current: arrival time, travel time, departure time, number of transfers, frequency, walking distance, used budget and cost. The Pareto-optimal routes can be traced back from the list of tuples of the arrival location. The average query time to perform multi-modal multi-criteria CSA is 225 ms (752 ms in the worst Valys case). In total this means that on overage 303 ms is needed to calculate all the possible routes from source to target when preprocessing is performed. Figure 10.2 summarizes the steps of the new journey planner, the last step is discussed in the next chapter. The frequency for a day can only be computed after the connections for that day are computed.

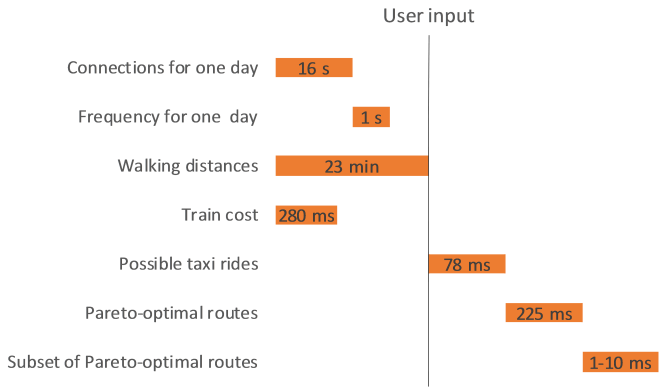


Figure 10.2: Chronological steps of the new journey planner. The first four steps are done before the user gives input (preprocessing) and the other three steps are done after the user gives input.

10.2. VALYS EXPERIMENTS

This section performs three experiments. The first experiment computes the average number of Pareto-optimal routes from source to target. The second experiments researches what the influence is on the travel time when being able to depart earlier or later. The third experiment researches what the influence is on the travel time when a restriction is put on a criterion.

All experiments are run on a laptop, having a 8GB memory and an Intel Core i7-3630QM 2.40GHz processor with 4 cores. All code is written in Java. The used data consist of Valys trips partly by train from last year which are delivered by Transvision [6] and all these trips are considered as Valys Vrij trips. The following conditions are taken into account: a walking speed of 4 km/h, a taxi price of €0.20 per km and a travel time for the taxi that is 1.4 times its fastest travel time (picking up other users requires extra travel time). The walking speed is based on a research done for the public transport of Londen [27] which assumes a maximum walking speed of 4.8 km/h. Lowering this to 4 km/h for the Valys target group seems reasonable, especially considering that most Valys users travel outside of peak hours as is determined in part I. If an even lower walking speed is required the user can give this as additional input. The taxi price and the maximum travel time by taxi are based on an agreement made between Transvision and the taxi companies.

The following restrictions are at least set on the criteria: at most 90 minutes more travel time partly by train than the travel time completely by taxi, at most 1000 meters of walking, at most 3 train transfers and at most 100 km by taxi. For all trips the departure moment is at 12:00 15-05-2017 (Monday). Unless stated otherwise these conditions apply to all experiments in this thesis. These restrictions are based on the following resources. Section 2.2 in part I showed that most Valys trips partly by train are between 12 to 84 minutes longer than Valys trips completely by taxi. Therefore a maximum of 90

minutes more travel time partly by train than the travel time completely by taxi is chosen. A research for the public transport of Londen [27] indicates a maximum walking distance of 960 m for rail, underground and light rail services. Given the target group of Valys it is chosen as the maximum walking distance for the complete trip. The maximum number of (train) transfers is set to 3 as this is the maximum number of (train) transfers that was done by a Valys user last year (2016). Of the Valys users who have used Valys Begeleid or Valys last year, 90% made 6 trips or less partly by train. Given a budget of 600 km per year the maximum number of taxi kilometers is set to 100 for a trip partly by train.

10.2.1. NUMBER OF PARETO-OPTIMAL ROUTES

For this experiment all Pareto-optimal routes are computed for Valys trips partly by train from last year. Figure 10.3 shows the number of Pareto-optimal routes if the trip is partly by train against the distance if the trip is completely by taxi. Three lines can be distinguished, each line has different restrictions set on the criteria. The orange line displays the number of Pareto-optimal routes when no restrictions are put on the criteria. This number of routes is way too large to show a user and keeps getting larger with the taxi distance (distance if the trip is completely by taxi and no other people are picked up). For the yellow line a couple restrictions are put on the criteria to filter out unreasonable routes. These restrictions include: at most 90 minutes more travel time partly by train than the travel time completely by taxi, at most 1000 meters of walking, at most 3 train transfers and at most 100 km by taxi. The line reaches its optimum at 100-125 km and gets relatively constant due to the restrictions set on the criteria. The same applies to the green line, which has an additional restriction of at most 50 km partly by train, only now the optimum is at 50-75 km. Setting restrictions on the criteria decreases the number of Pareto-optimal routes drastically, but the number of Pareto-optimal routes is still too large to show a user.

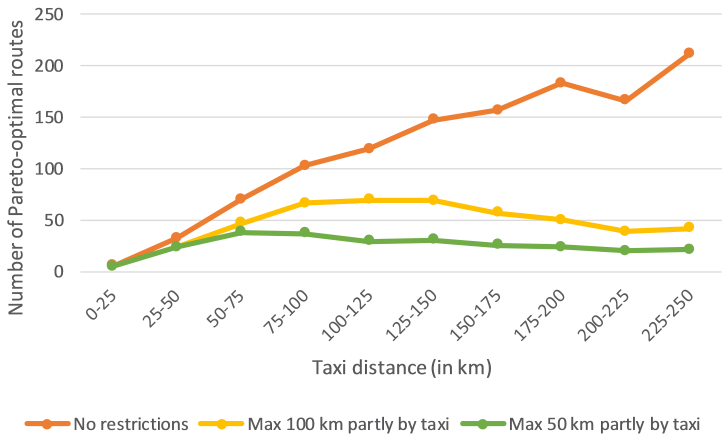


Figure 10.3: Number of Pareto-optimal routes for different restrictions set on the criteria

10.2.2. DEPARTING EARLIER OR LATER

This section researches the influence on the travel time when being able to departure earlier or later than the given departure time. Figure 10.4 displays the average gain in travel time, over all Pareto-optimal routes, when being able to departure 5, 10 or 15 minutes later. The gain in travel time is caused by shorter waiting times as a result of the possibility to take a taxi or train earlier or later. Being able to depart 15 minutes earlier or later does not yield much additional gain (less than one minute on average) when comparing it to being able to depart 10 minutes earlier or later, therefore in the next experiments the amount of time the user can departure earlier or later is set to 10 minutes. Allowing the user to depart 10 minutes earlier or later results on average in a gain of 8 minutes.

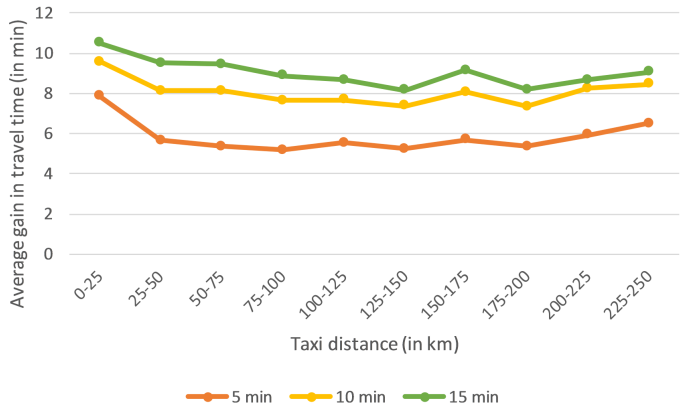


Figure 10.4: Average gain in travel time when being able to depart 5, 10 or 15 minutes earlier or later

10.2.3. INCREASE IN TRAVEL TIME

For this experiment the same conditions apply as in the previous section, only the restrictions set on the criteria differ. This section researches the influence of setting (additional) restrictions on the criteria: number of transfers, frequency, walking distance and budget. The travel time of the Pareto-optimal route with the fastest travel time under a set of restrictions is chosen for every trip.

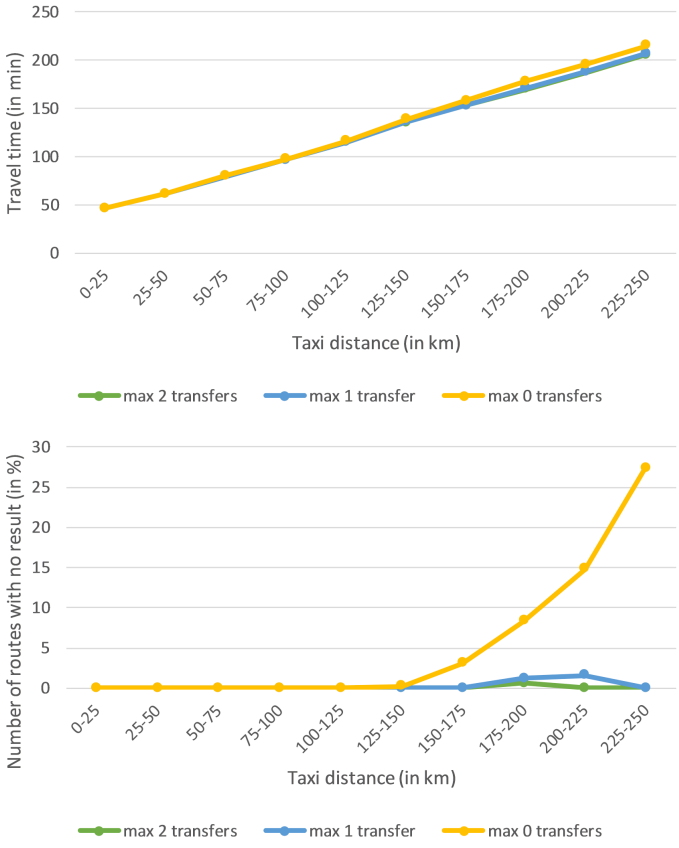


Figure 10.5: Influence of putting a restriction on the number of transfers

Figure 10.5 displays what happens when additional restrictions are set on the number of transfers. The travel time barely changes, this means that the fastest (or almost fastest) route already has zero train transfers. In the most extreme case there is a travel time difference of 9 minutes between 0 and 2 transfers. When the distance of the trip is larger than 150 km it is possible that there exist no route with at most 0 or 1 transfers. Next figure 10.6 displays what happens when additional restrictions are set on the frequency. Again the travel time barely changes, in the most extreme case there is a travel time difference of 14 minutes between a minimum of 2 and 4 in frequency. Setting a restriction on the frequency does have a high influence on the number of trips without an existing route, especially for very short and very large distance there does not always exist a route with a minimum of 3 or 4 in frequency.

Subsequently figure 10.7 displays what happens when additional restrictions are set on the walking distance. The difference in travel time is larger than for the previous two criteria restrictions, in the most extreme case there is a travel time difference of 23 min between a walking distance of 250 m and 1000 m. When the distance of the trip is larger

than 100 km it is possible there exists no route with at most 250 m of walking. At last figure 10.8 displays what happens when additional restrictions are set on the budget. In the most extreme case there is a travel time difference of 16 min between 25 and 100 km in budget. When the distance of the trip is larger than 75 km, is is possible there exist no route which spends at most 25 km of the budget.

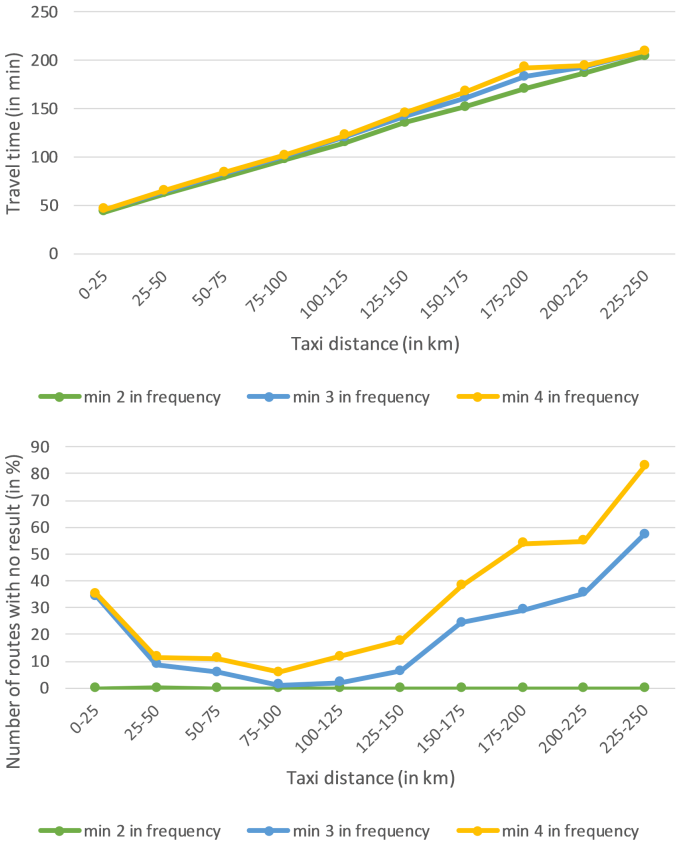


Figure 10.6: Influence of putting a restriction on the frequency

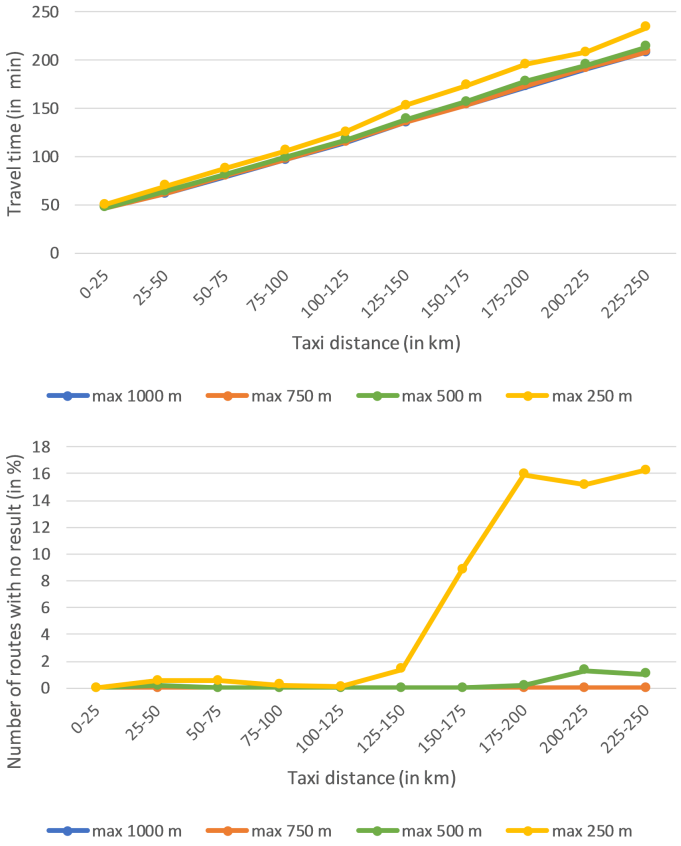


Figure 10.7: Influence of putting a restriction on the walking distance

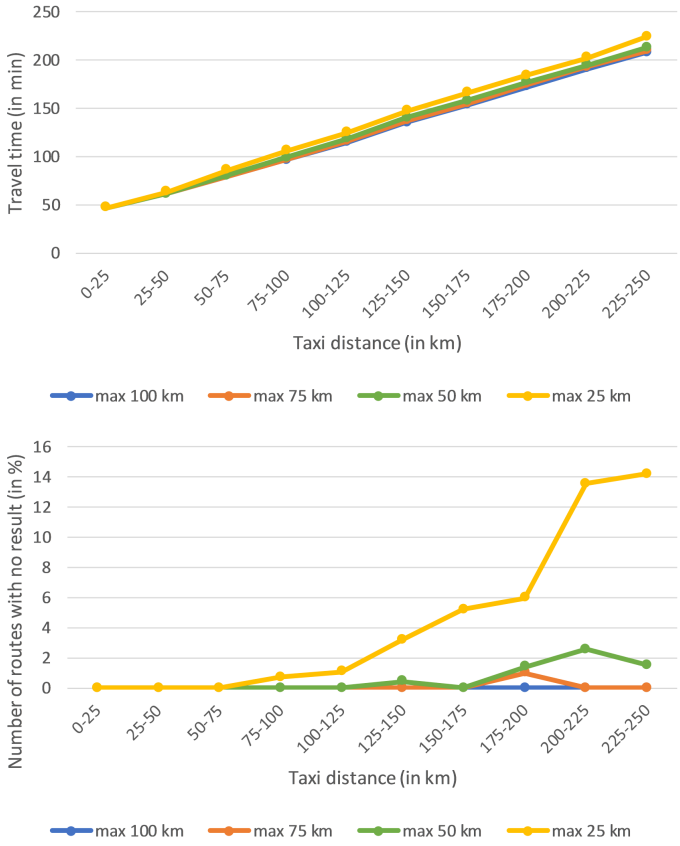


Figure 10.8: Influence of putting a restriction on the budget

10.2.4. CONCLUSION

The experiments in the previous sections lead to the following conclusions. The calculated number of Pareto-optimal routes are too large too show a user even after filtering out unreasonable routes by setting restrictions on the criteria. Being able to depart 10 minutes earlier or later is found to be most suitable and results on average in a travel time gain of 8 minutes. The highest increase in travel time is when additional restrictions are set on the walking distance. The highest number of journeys with no result in percentage are when additional restrictions are set on the frequency.

11

SUBSET OF PARETO-OPTIMAL JOURNEYS

The previous chapter concluded that the computed number of Pareto-optimal journeys is too large to show a user, even after filtering out the unreasonable journeys. This chapter discusses how a representative subset of journeys can be chosen from the set of Pareto-optimal journeys. Section 11.1 defines the requirements for this subset and section 11.2 introduces an example journey that is used in the rest of this chapter. Next section 11.3 discusses three methods to select a subset of representative journeys. Each method has a different mechanism to score the journeys and only the top k is meant to be shown to a user. Section 11.4 compares the three methods on Valys trips partly by train from last year (2016). At last section 11.5 describes how the user can input his or her routing preferences and section 11.6 explains how findings from part I can be used in selecting suitable subsets.

11.1. REQUIREMENTS

Since the Pareto-optimal set of journeys contain all the non-dominated journeys, each journey is important as it is a trade-off of different criteria. The user should be shown a subset of Pareto-optimal journeys. The reason for not showing just one journey is that the exact trade-offs of the criteria can not be known, because it is about people and people can change their opinion very easily. The trade-offs of the criteria can also be different per journey and is not always linear (e.g. 3 transfers can be much worse than 2 transfers).

By giving a subset of journeys, instead of just one journey, the user can make the last trade-off himself or herself. Without knowing an approximation of the trade-offs of the criteria it is hard to determine which subset of journeys are best to show a user. In that case a representative subset should be shown, which contains a good journey for (almost) anyone. When in a later stage more is known about the routing preferences of the user, the subset can be based on those preferences.

11.2. EXAMPLE JOURNEY

This section introduces a journey that is used in the next section as an example Valys trip to explain the three different methods that select a subset of representative journeys. For privacy reasons a non-existing Valys journey is chosen. A journey is chosen for which a long distance has to be traveled, several (both large and small) stations are nearby, as well as some major roads so that the number of Pareto-optimal journeys that follow are high. The journey is from Woerdsense Verlaat (A) to Zwartsluis (B) as is displayed in figure 11.1. The distance completely by taxi is 130 km and the travel time completely by taxi is at most 128 minutes (this includes picking up other Valys users).

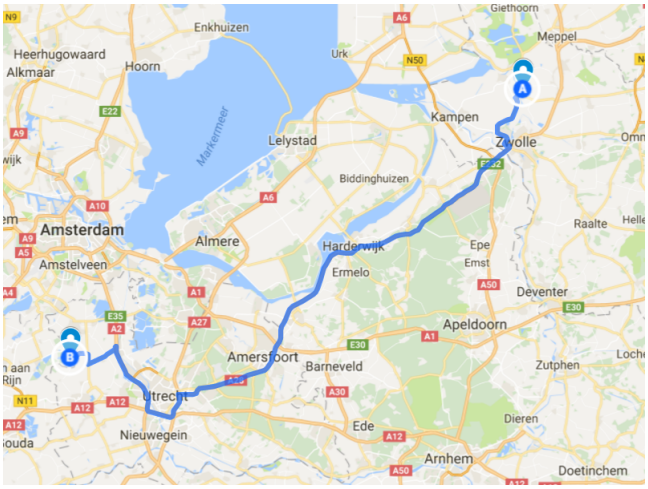


Figure 11.1: Journey from Woerdense Verlaat (A) in Overrijsel to Zwartsluis (B) in Zuid-Holland. The line portrays the shortest journey completely by taxi. The map is made with the My Maps tool of Google Maps [4]

.

The solution from the previous chapter is used to compute all Pareto-optimal journeys partly by train from A to B, again the following six criteria are taken into account: travel time, number of transfers, frequency, walking distance, budget and cost. To gain reasonable journeys partly by train a couple restrictions are put on the criteria: at most 90 minutes more travel time partly by train than the travel time completely by taxi, at most 1000 meters of walking, at most 3 train transfers and at most 100 km by taxi. This leaves 63 Pareto-optimal journeys.

11.3. SUBSET METHODS

11.3.1. WEIGHTED SUM METHOD

A multi-objective ranking method can be used to rank the Pareto-optimal set. A classical multi-objective ranking method is the weighted sum method:

$$\sum_{i=1}^{n} w_i f_i \tag{11.1}$$

where n is the number of objectives, f_i is the objective function of objective i and w_i is the weight of objective i as is described by K. MacCrimmon [28]. This method can be applied to routing, by computing for every journey the weighted sum and then ranking the journeys on the weighted sum value. The values of the criteria for every journey need to be normalized beforehand.

There are a two problems with this method:

- The subset of journeys are not always a good representation of the Pareto front
- The trade-offs of the criteria need to be known (values for the weights)

In the example Valys trip six criteria are taken into account, figure 11.2 shows the weight vector for this (in the left matrix). For example a weight vector of [1 1 0 0 0 0] means that the user finds the criteria travel time and the number of transfers equally important, but the other criteria not important.

$$W = \begin{bmatrix} w_1\text{: travel time} \\ w_2\text{: number of transfers} \\ w_3\text{: frequency} \\ w_4\text{: walking distance} \\ w_5\text{: budget} \\ w_6\text{: cost} \end{bmatrix} \begin{bmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 1&0&0&0&0&1&1&1&1&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&0&0&1&1&1&0&0&0 \\ 0&0&1&0&0&0&1&0&0&1&0&0&1&1&0 \\ 0&0&0&1&0&0&0&1&0&0&1&0&1&0&1 \\ 0&0&0&0&1&0&0&0&1&0&0&1&0&1&1 \end{bmatrix} \tag{11.2}$$

Table 11.1 displays the top five journeys when the weighted sum method with weight vector [1 1 0 0 0 0] is applied to the example Valys trip. These five journeys are very similar and none of the journeys have a transfer by train. Even though the subset is not a good representation of the Pareto front, it is a good set of journeys for a Valys user that is only interested in travel time and number of transfers.

Table 11.1: Top five journeys for the example Valys trip. The weighted sum method is used to rank the journeys based on a weight vector of [1 1 0 0 0 0].

journey	travel time (in min)	number of transfers	frequency	walking distance (in m)	budget (in km)	cost (in €)
1	**133**	0	2	739	**42**	**23,7**
2	138	0	2	294	59	23,9
3	140	0	2	162	51	**23,7**
4	141	0	2	525	47	27,6
5	143	0	2	**112**	97	27,4

Not only the travel time and number of transfers can be of interest to the Valys user, but also other criteria. Based on findings from part I the most common weight vectors can not be determined, but it is possible to roughly estimate the weight vectors. Travel

time is important to each user in a certain form. In general the Valys user finds travel time in combination with one or two criteria important. When only equal weights are taken into account for the criteria, the corresponding weight vectors are as is displayed in the right matrix of figure 11.2. Next the top five journeys for every weight vector are computed. Table 11.2 displays the top five most common journeys for these weight vectors and figure 11.2 shows a visualization of these fives journeys on a map.

Table 11.2: Top five most common journeys based on using multiple weight vectors for the weighted sum method

journey	travel time (in min)	number of transfers	frequency	walking distance (in m)	budget (in km)	cost (in €)
1	**133**	**0**	2	739	42	23,7
2	140	**0**	2	162	51	23,7
3	138	**0**	2	294	59	23,9
4	144	1	2	707	**28**	**23**
5	145	**0**	2	**100**	62	24,5

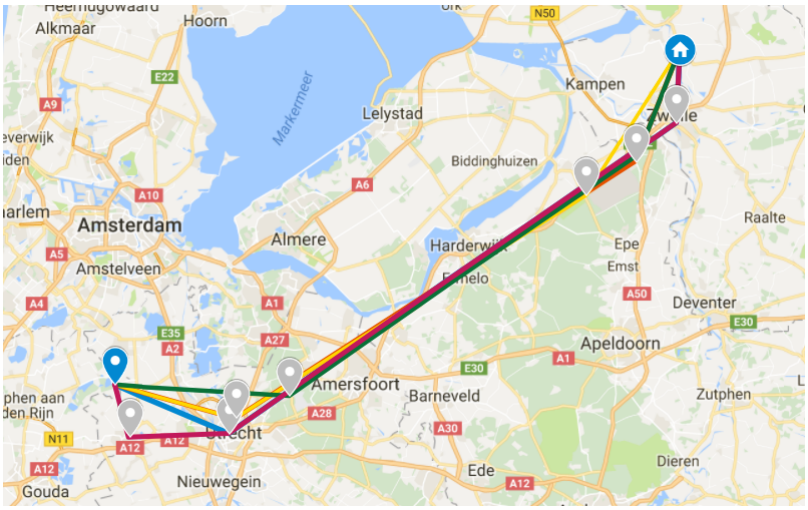


Figure 11.2: Visualization of the top five journeys chosen by the weighted sum method. The map is made with the My Maps tool of Google Maps [4].

Most weight vectors got their number one journey in the top five most common list, this includes the following weight vectors:

- [1,1,0,0,0,0] (travel time, number of transfers)
- [1,0,0,0,1,0] (travel time, budget)
- [1,0,0,0,0,1] (travel time, cost)

- [1,1,0,0,1,0] (travel time, number of transfers, budget)
- [1,1,0,0,0,1] (travel time, number of transfers, cost)
- [1,0,1,0,1,0] (travel time, frequency, budget)
- [1,0,0,1,1,0] (travel time, walking distance, budget)
- [1,0,0,1,0,1] (travel time, walking distance, cost)
- [1,0,0,0,1,1] (travel time, budget, cost)

Next the following weight vectors got their third best journey in the top five most common list :

- [1,0,0,1,0,0] (travel time, walking distance)
- [1,1,0,1,0,0] (travel time, number of transfers, walking distance)

Furthermore the following weight vectors got their fifth best journey in the top five most common list:

- [1,1,1,0,0,0] (travel time, number of transfers, frequency)

At last the following weight vectors got non of their five best journeys in the top five most common list:

- [1,0,1,0,0,0] (travel time, frequency)
- [1,0,1,1,0,0] (travel time, frequency, walking distance)
- [1,0,1,0,0,1] (travel time, frequency, cost)

This indicates that for the example Valys trip that even when the weight vector is not known for the user (and the specific journey) it is still possible to show a selection of journeys that has at least one good match for most users. Apart from that a journey with a higher frequency is missing, the rough estimate of the weight vectors results in a good representation of the Pareto front for the example Valys trip.

The weights being partly unknown for a user could cause for some users that his or her best journey(s) are not shown. When the number of journeys shown to the user are lowered, the chance of it including a good match for the current user is also lower. When the user starts making more trips, the weight vector(s) can be better approximated based on previously chosen journeys, this results in a subset of journeys that are better adjusted to the preferences of the user.

11.3.2. CROWDING DISTANCE

This section discusses a method that does not require weights. The evolutionary algorithm NSGA-II uses a method called the crowding distance to sort the solutions of the Pareto front and this algorithm is introduced by K. Deb et al. [29]. The crowding distance method is used to preserve diversity among solutions in the same non-dominated front and is meant for general multi-objective optimization problems. Diversity is something

that is required in selecting a subset of journeys of the Pareto-optimal set. When using the crowding distance method to rank the journeys it works as follows. The crowding distance is calculated separately for every criterion:

- The journeys are sorted on the current criterion
- The first and last journey after sorting is assigned a crowding distance of infinity
- The crowding distance for the other journeys are defined as the normalized difference between the criterion value of the next journey and criterion value of the previous journey

The crowding distance for a journey is equal to the sum of the crowding distances for every criterion. Assigning the first and last journey a value of infinity for the crowding distance could cause journeys that are only good in one criterion to have a too high crowding distance value. To eliminate this the crowding distance for the first journey is the difference between the first and the second journey for the current criterion. The crowding distance for the last journey is the difference between the last journey and the journey before that for the current criterion.

Table 11.3 shows the top five journeys when the Pareto-optimal journeys of the example Valys trip are sorted on crowding distance and figure 11.3 shows the visualization of these five journeys on a map. It has one journey in common with the top five journeys of the weighted sum method. Other than, the crowding distance method shows a worse representation of the Pareto front for the example Valys trip than the weighted sum method. The minimum value of the top 5 for the walking distance, budget and cost is lower for the minimum weighted method than for the crowding distance method. Still the top 5 journeys of the crowding distance need more travel time and sometimes more transfers. The crowding distance aims to show a diversity of solutions of the Pareto front (figure 11.3 emphasizes this), but at the same time the journeys are extremely worse off in one or multiple criteria for the example Valys trip.

Table 11.3: Top five journeys for the example Valys trip after sorting on crowding distance.

journey	travel time (in min)	number of transfers	frequency	walking distance (in m)	budget (in km)	cost (in €)
1	181	3	**4**	823	87	24.9
2	154	**0**	3	372	85	27.8
3	**133**	**0**	2	739	42	**23.7**
4	212	2	3	**246**	79	31.3
5	214	1	2	468	**34**	26.3

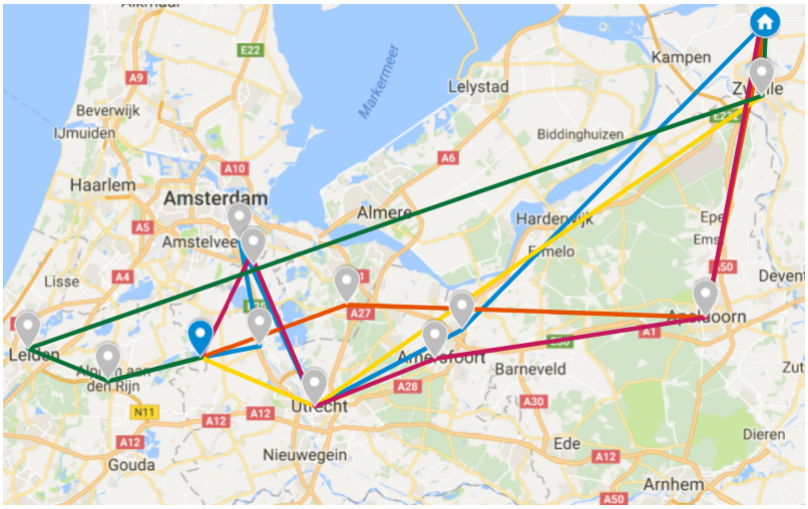


Figure 11.3: Visualization of the top five journeys chosen by the crowding distance method. The map is made with the My Maps tool of Google Maps [4].

11.3.3. FUZZY DOMINANCE

The fuzzy dominance method, as is described in section 7.3, is the only current method in literature that chooses a subset of journeys by more than just filtering out unreasonable journeys. For the Valys case the values for the Gaussian function are chosen as is displayed in table 11.4 and are based on (χ,ϵ) values as described by D. Delling et al. [19].

The values for the Gaussian function for the criteria number of transfers and walking distance are exactly as the ones described by D. Delling et al [19]. The values for the Gaussian function for the criterion travel time are a bit less strict than described by Delling et al., (0.8,2) instead of (0.8.1). The values for the Gaussian function for the frequency criterion are set the same as for the number of transfers criterion and the values for the Gaussian function for the budget criterion are set the same as for the travel time criterion, because the variance of these criteria are similar. At last the cost criterion is set to (0.8,1) instead of the (0.8,5) as described by Delling et al. Gaussian values of (0.8,5) view a difference of 20 euros in cost as big and Gaussian values of (0.8,1) view 5 euros already as a big difference and this is better suited in the context of Valys.

Originally the score for a journey is based on the maximum domination degree, but for the Valys case this is changed to the sum of the domination degrees of all other journeys, otherwise more than 5 journeys could have a score of 1.

Table 11.4: Gaussian function values for every criterion

	χ	ϵ
travel time	0.8	2
number of transfers	0.1	1
frequency	0.1	1
walking distance	0.8	5
budget	0.8	2
cost	0.8	1

Table 11.5 shows the top fives journeys when the fuzzy dominance method is applied and figure 11.4 shows a visualization of these five journeys on a map. When comparing this with the top five journeys of the weighted sum method for the example Valys trip four out of five journeys are the same. The weighted sum method has one journey that is both cheaper and uses less budget than all other journeys in the top five of the fuzzy dominance method. For the example Valys trip the weighted sum method chooses a similar, but slightly better representation of the Pareto front. The fuzzy dominance method does chooses a far better representation of the Pareto front than the crowding distance method. Figure D.1 in appendix D visualizes the chosen subsets for each of the three methods on the example Valys trip.

Table 11.5: Top five journeys for the example Valys trip based on the fuzzy dominance method

journey	travel time (in min)	number of transfers	frequency	walking distance (in m)	budget (in km)	cost (in €)
1	140	0	2	162	51	**23.7**
2	138	0	2	294	59	23.9
3	145	0	2	100	62	24.5
4	**133**	0	2	739	**42**	**23.7**
5	144	0	2	**95**	89	26.5

Figure 11.4: Visualization of the top five journeys chosen by the fuzzy dominance method. The map is made with the My Maps tool of Google Maps [4].

11.4. COMPARING THE GENERATED SUBSETS

This section compares the top five chosen journeys obtained by the weighted sum method, the crowding distance method and the fuzzy dominance method on Valys trips partly by train from last year (2016). These methods are compared on the following metrics:

1. Runtime on average and runtime in the worst case scenario.
2. Best distance to the minimum. The difference between the minimum value of the top 5 journeys and the minimum value of all journeys for a criterion.
3. Average distance to the minimum. The difference between the average value of the top 5 journeys and the minimum value of all journeys for a criterion.
4. Worst distance to the maximum. The difference between the maximum value of the top 5 journeys and the maximum value of all journeys for a criterion.
5. Average distance to the maximum. The difference between the average value of the top 5 journeys and the maximum value of all journeys for a criterion.
6. Diversity. The standard deviation of the top five journeys for a criterion.

These metrics are calculated for every criterion, every journey and every method. An ideal method should have a low runtime, a small distance to the minimum, a high distance to the maximum and a high diversity for every criterion. Figure 11.5, 11.6, 11.7, 11.8, 11.9 and 11.10 display the results for the last 5 metrics averages over all Valys trips partly by train from last year (2016). Each figure shows the metrics for a different criterion.

The crowding distance method is the method with the largest diversity, especially for the criteria travel time and cost. At the same time the crowding distance method is the method with criteria values closest to the maximum. This results in unwanted trade-offs where one or multiple criteria are too high.

This leaves the weighted sum method and the fuzzy dominance method. There is no clear winner following from the column graphs. The weighted sum method has a higher diversity for five out of six criteria. The results for the (average) distance to the minimum and (average) distance to the maximum show that the weighted sum method is for most criteria slightly closer to the minimum, but the fuzzy dominance method is for most criteria slightly farther away from the maximum. The runtime of the different methods are compared in table 11.6. Relatively seen the weighted sum method is a lot faster than the fuzzy dominance method, this is because the fuzzy dominance method compares each journey with every other journey. Because the weighted sum method achieves similar results to the fuzzy dominance method, but is much faster it is chosen as the best method to choose a subset of journeys in the context of Valys.

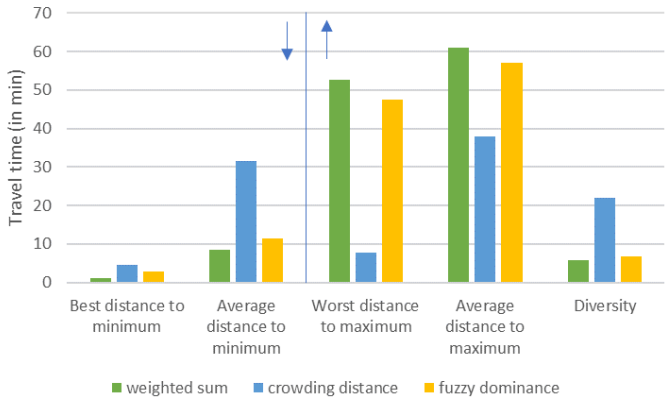


Figure 11.5: Comparison of the three subset methods on the criterion travel time. The left 2 metrics need to be as low as possible and the right 3 metrics need to be as high as possible.

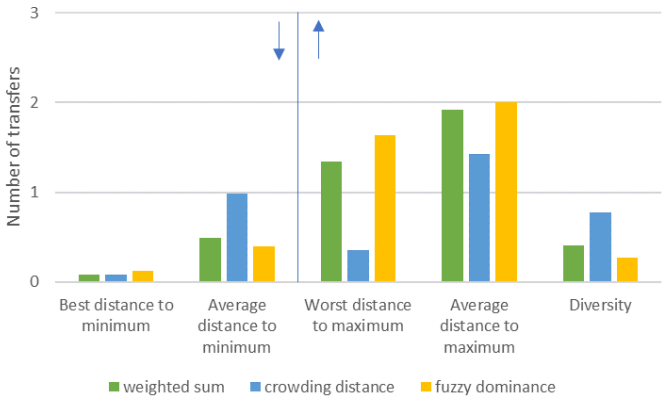


Figure 11.6: Comparison of the three subset methods on the criterion number of transfers. The left 2 metrics need to be as low as possible and the right 3 metrics need to be as high as possible.

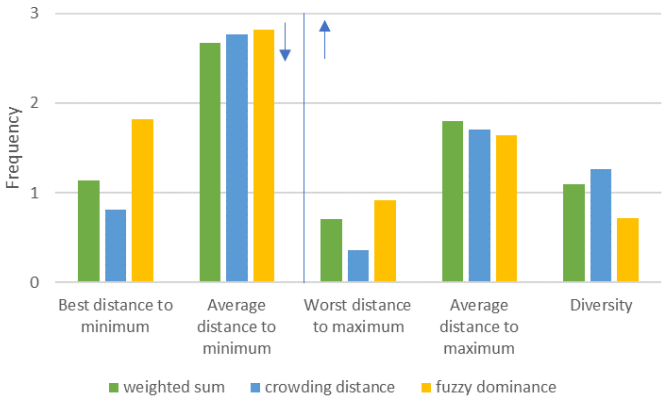


Figure 11.7: Comparison of the three subset methods on the criterion frequency. The left 2 metrics need to be as low as possible and the right 3 metrics need to be as high as possible.

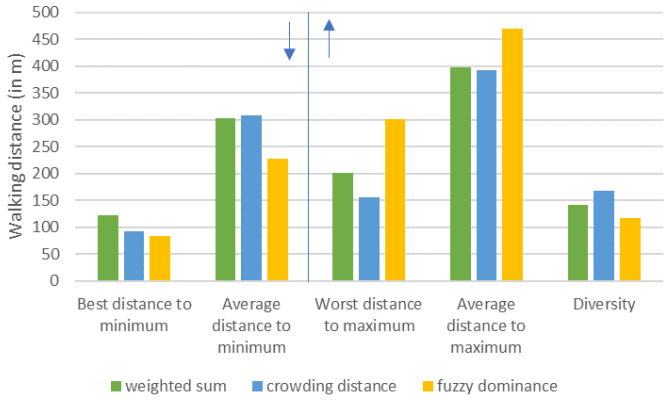


Figure 11.8: Comparison of the three subset methods on the criterion walking distance. The left 2 metrics need to be as low as possible and the right 3 metrics need to be as high as possible.

.

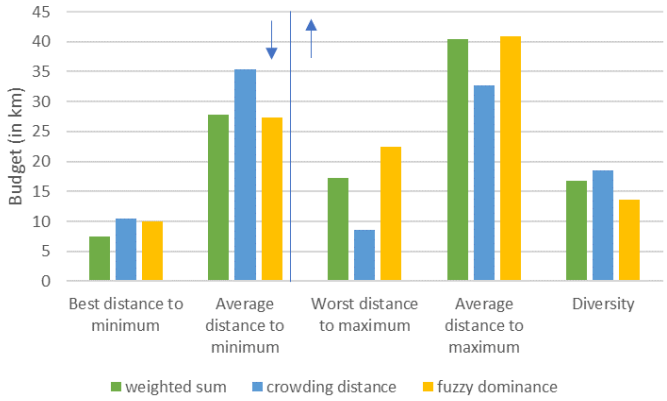


Figure 11.9: Comparison of the three subset methods on the criterion budget. The left 2 metrics need to be as low as possible and the right 3 metrics need to be as high as possible.

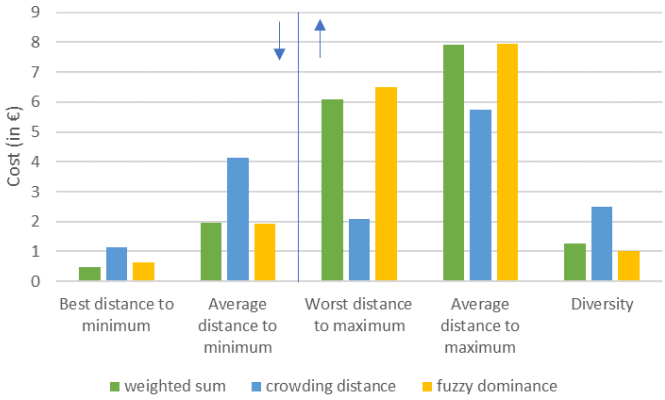


Figure 11.10: Comparison of the three subset methods on the criterion cost. The left 2 metrics need to be as low as possible and the right 3 metrics need to be as high as possible.

Table 11.6: Runtime comparison for the methods: weighted sum, crowding distance and fuzzy dominance

	runtime on average (in ms)	runtime in worst case (in ms)
weighted sum	2	24
crowding distance	1	14
fuzzy dominance	10	162

11.5. USER INPUT

It is also possible to let the user input his or her routing preferences and this section covers how that can be achieved. Not every Valys user is capable of indicating his or her routing preferences, for those that are not capable the representative subset generated by the weighted sum method or the fuzzy dominance is more suitable.

Allowing the user to set restrictions on the criteria can already reduce the set of journeys to a list of journeys that is found to be reasonable to the user. For the following criteria an absolute value can be used as a restriction: number of transfers, frequency, walking distance and budget. For the criteria time and cost it is better to make a restriction based on a comparison with the trip completely by taxi. For example by only allowing trips partly by train that require at most 90 minutes more travel time than the trip completely by taxi.

Figure 11.11 displays how the user can indicate how important he or she finds every criterion, this can be converted to weights between 0 and 1 for every criterion and this results in a weight vector that can be used for the weighted sum method. For both the crowding distance method and the fuzzy dominance method weights are not required, but it is possible to pass these in the equation. At last the user can be even more in control when he or she can decide to which and from which train station he or she goes by taxi. Additionally the user can indicate a "through train station x" option.

The design in figure 11.11 is based on a journey planner designed by Andre et al. [30]. In this journey planner the user could express the importance to avoid bad weather, avoid areas known for posing a risk of criminal activity, avoid public transport at peak times, avoid unlit areas (parks or roads without street-lightning), travel for as little time as possible, travel for as low in cost as possible and/or exercise (such as walking). This journey planner was only designed in UI form, the back end of the journey planner was never created.

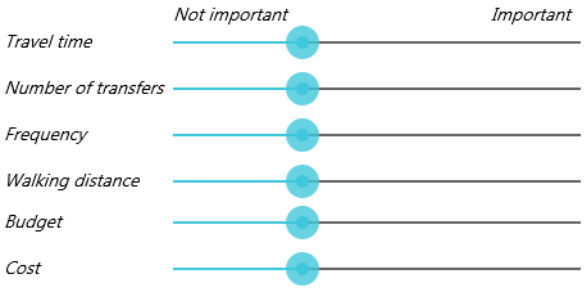


Figure 11.11: Weights chosen by the user (designed with Mockplus[5])

11.6. INFLUENCE OF FINDINGS FROM PART 1

A number of findings from the first part of this thesis can play a role in the subset of journeys that should be shown to the user. First of all most Valys users (54%) only make 1 to 4 trips a year with Valys. For these users optimization on budget seems unnecessary. For example travel time and number of transfers are more important criteria here.

Some of the current users indicated not to travel partly by train, since they do not want to make a train transfer. It would be a good idea to show users who have never travelled partly by train at least one journey where no train transfers are needed, this is important to lower the threshold to travel partly by train.

For trips larger than 70 km made by users of relatively young age (around 65) and who spend most of their budget there is a high change that these trips will be made by train. This group of users makes relatively seen a lot of trips and therefore budget is an important factor. This group may also manage to take one or more train transfers due to their experience in travelling partly by train.

The moment of the trip and the amount of budget the user has left should also influence the subset of journeys shown to the user. For example when the user already used half of his or her budget, but it is still the beginning of the year, the user should be shown journeys that do no require a lot of budget.

12

COMPARISON WITH THE VALYS JOURNEY PLANNER

This section compares the current journey planner of Valys with the journey planner designed in this thesis. Section 12.1 gives a general comparison between the two journey planners. Next section 12.2 and 12.3 give for each journey planner a definition for best route. At last section 12.4 compares the journey planners based on Valys trips partly by train from last year (2016).

12.1. GENERAL COMPARISON

This section gives a general comparison between the journey planner of Valys and the journey planner designed in this thesis. The new journey planner is partly based on the limitations of the Valys journey planner as is described in section 8.2. The following list summarizes the improvements of the new journey planner over the Valys journey planner:

- It takes all interesting train stations into account to drive to or from by taxi and does not limit itself to only the three nearest train stations from the departure location or to the arrival location.
- It optimizes on many criteria, not just on travel time. These criteria include: travel time, number of transfers, frequency, walking distance, budget and cost.
- It can take restrictions for the criteria into account.
- The needed transfer time is based on the actual walking distance and the walking speed of the user. The walking speed is given as input.
- It computes as much as possible beforehand. The Valys planner computes everything after the user gives his or her user input

- It can take trade-offs for the criteria given by the user into account to create a top x routes. Without such trade-offs for the criteria a representative set of journeys is shown. The Valys journey planner always shows the fastest, cheapest and best route, but these could all be the same route.
- It shows journeys than can depart earlier or later than the departure time given by the user.

12.2. BEST ROUTE - VALYS JOURNEY PLANNER

This section explains how the Valys journey planner is imitated to compute the best route. First the Travel Matrix of Calendar42 [26] is used to compute the distance and time to the three nearest train stations from the departure location by taxi. Also the distance and time from the three nearest train stations to the arrival location by taxi are computed with the Travel Matrix.

Given the number of train stations taken into account to drive to or from, there are nine possible routes from the departure location to the arrival location. CSA is used to compute these nine routes, optimization on travel time takes place, if there exist a route that has the same arrival time at the destination, but with fewer transfers then this route is chosen. The transfer time that Valys currently takes into account is delivered by Transvision [6]. Two transfer times per train station are defined, one for transferring between trains and another for transferring between a taxi and a train. The best route defined by Valys (out of the nine routes) is the route with the lowest value for the summation of:

- The number of taxi kilometers
- The total distance (taxi in combination with train) × 0.20
- The total travel time × 0.05

12.3. BEST ROUTE - NEW JOURNEY PLANNER

To compare the Valys journey planner with the new journey planner, a definition for the *best* route is also needed for the new journey planner. The weighted sum method as is described in section 11.3.1 is used with a weight vector of [1 1 0 0 1 1]. This means the criteria travel time, number of transfers, budget and cost are found to be equally important. On the other hand the criteria frequency and walking distance are not found to be important. The decision to not include frequency and walking distance in the trade-off is because these are also not computed by the Valys journey planner.

Just as in other sections, to gain reasonable routes, a couple restrictions are put on the criteria: at most 90 minutes more travel time partly by train than the travel time completely by taxi, at most 1000 meters of walking, at most 3 train transfers and at most 100 km by taxi. The following conditions are taken into account: a walking speed of 4 km/h, a taxi price of €0.20 per km and a travel time for the taxi that is 1.4 times it fastest travel time (picking up other users requires additional travel time).

12.4. COMPARING BEST ROUTES

This section compares the best route of the Valys journey planner with the best route of the journey planner designed in this thesis. The comparison is done on Valys trips from last year, again only the trips that are partly by train are taken into account and all trips are considered to be Valys Vrij trips. For all trips the departure moment is at 12:00 15-05-2017 (Monday).

On average the new journey planner spends 12.5 km more on budget and €0.5 more in cost, but gains on overage 51.3 minutes in travel time and 1.1 in transfers. Figure 12.1, 12.2, 12.3 and 12.4 compare the best routes on travel time, number of transfers, budget and cost. Figure 12.1 shows a linear growth in travel time for both journey planners and the difference in travel time is relatively constant. The gain in travel time for the new journey planner is caused by: allowing the user to depart earlier or later than the given departure time, walking distance dependent transfer time and allowing to drive longer by taxi. The additional line on the graph indicates the travel time completely by taxi and shows that the new journey planner is much closer to this line than the Valys journey planner.

By allowing to drive longer by taxi less transfer are needed (this also means less transfer time) and the taxi is (even when picking up multiple people) still most of the time faster than the train. For large distances the gain in amount of transfers can take up to 2 transfers as is displayed in figure 12.2. The budget use becomes more or less constant for trips larger than 75 km as is deducible from figure 12.3. Finally the cost, as is displayed in figure 12.4, hardly differs between the two journey planners.

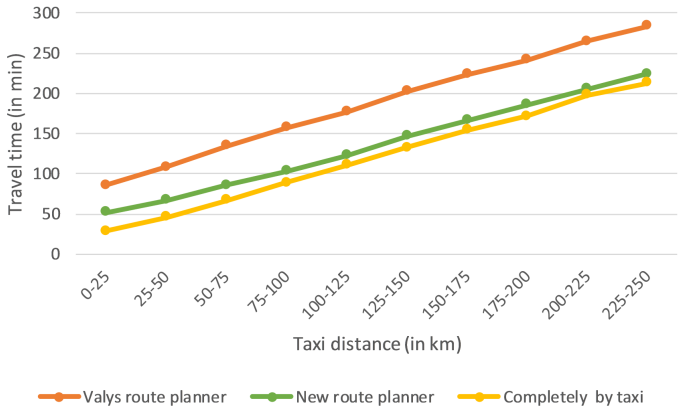


Figure 12.1: Comparison on travel time. The best route of the Valys journey planner is compared with the best route of the new journey planner.

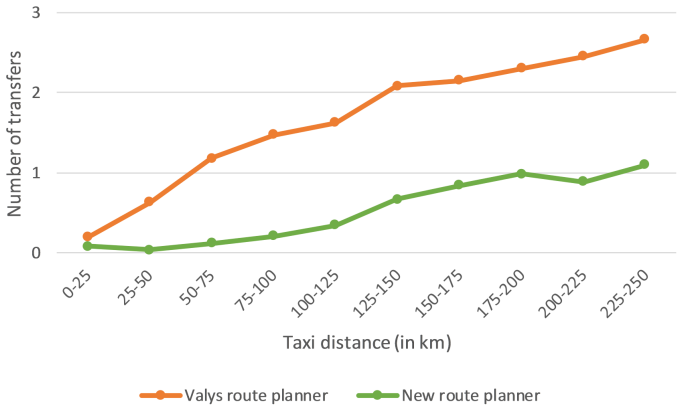


Figure 12.2: Comparison on number of transfers. The best route of the Valys journey planner is compared with the best route of the new journey planner.

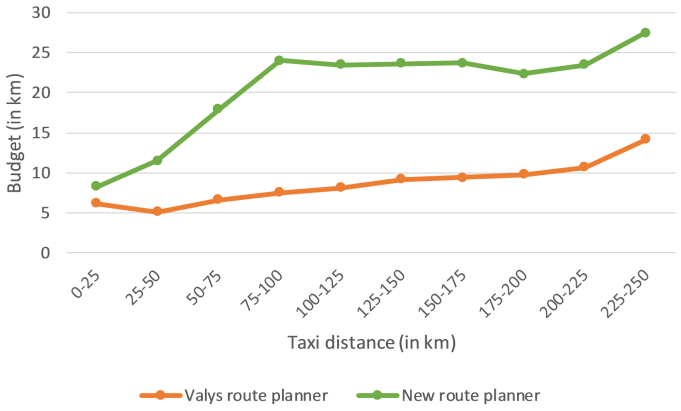


Figure 12.3: Comparison on travel budget. The best route of the Valys journey planner is compared with the best route of the new journey planner.

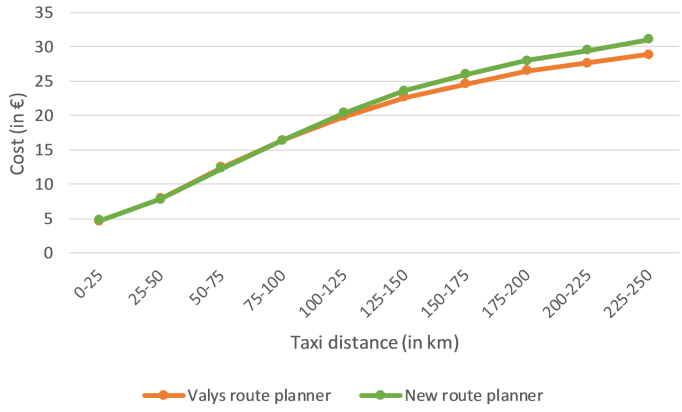


Figure 12.4: Comparison on cost. The best route of the Valys journey planner is compared with the best route of the new journey planner.

13

CONCLUSION

The goal of part II of this thesis was to create a better and more personalized journey planner for Valys in such as way that it could apply to more applications than just Valys by using observations from part I. The research question "How can a representative set of routes be found that is not too big from source to target that combines multiple transportation modes, takes into account many criteria, can take into account restrictions, is fast enough and focuses first on one unrestricted transportation mode, then one or multiple schedule-based transportation modes and at last one unrestricted transportation mode?" is answered in steps.

First all Pareto-optimal routes from source to target are computed with CSA by converting the unrestricted and schedule-based transportation modes to connections, in this way CSA is combined with existing techniques developed for road networks. Every stop stores a list of non-dominated tuples, where each tuple contains the current values for the criteria, the last connection and a reference to the previous tuple. The following criteria are taken into account: travel time, number of transfers, frequency, walking distance, budget and cost. All possible values for the frequency, train cost and walking distance are already computed before the user plans his or her journey. The resulting algorithm is fast enough, but returns a large set of Pareto-optimal routes even after filtering out unreasonable routes.

When a Valys user is able to depart 10 minutes earlier or later than the given departure time this gains on average 8 minutes in travel time. When a Valys user is able to depart 15 minutes earlier or later this gains only one minute extra compared to departing 10 minutes earlier or later. The highest increase in travel time is when additional restrictions are set on the walking distance. The highest number of routes with no result in terms of percentage are when additional restrictions are set on the frequency.

Next three methods to select a subset of representative routes are discussed and compared. Currently the fuzzy dominance method is the only method in literature that selects a subset of Pareto-optimal routes by more than just filtering out unreasonable routes. The crowding distance method, based on the sorting algorithm used by evolutionary algorithm NSGA-II, gives a worse subset of Pareto-optimal routes compared

to the weighted sum and the fuzzy dominance method. As the weighted sum method achieves similar results to the fuzzy dominance method, but is much faster it is chosen as the best method (out of the three) to choose a subset of routes in the context of Valys. When the user is capable of giving trade-offs for the criteria, then the weighted sum method (or fuzzy dominance method) can be used to rank the routes based on this. After the routes are ranked, only the top x routes are shown to the user.

At last the current journey planner of Valys is compared with the journey planner designed in this thesis. The journey planner designed in this thesis takes into account all potential train stations to drive to or from by taxi, minimizes on many criteria, takes into account restrictions, uses a transfer time based on actual walking distance and walking speed of the user, can take trade-offs for the criteria into account and uses preprocessing. By spending on average 12.5 km more on budget and €0.5 more in cost it is possible to gain on average 51.3 minutes in travel time and 1.1 in transfers with the new journey planner. The new journey planner is in travel time much closer to the trip completely by taxi compared to the Valys journey planner.

III

FUTURE WORK AND RECOMMENDATIONS

14

FUTURE WORK AND RECOMMENDATIONS

For future work multiple areas are of interest. First of all the the journey planner designed in this thesis does not take into account delays or cancellations of trains. Dibbelt et al. [31] introduce an algorithm called MEAT that computes back up journeys for the connections of CSA. When a journey is delayed, this can effect the values for the criteria of some of the Pareto-optimal routes and possibly other journeys dominate the currently chosen journey. The best way to solve this needs to be researched. The chance of a train delaying could also be added as an additional criteria to optimize on.

Next the precise input of the Valys user needs to be determined. Is giving the user the option to put restrictions on the criteria and to select trade-offs for the criteria too much or still suitable for some users? Also the best representation of the routes and the maximum number of routes suitable to show a user need to be determined.

Furthermore taking other schedule-based transportation modes into account besides trains will most likely slow the algorithm. By storing not all Pareto-optimal routes, but making the selection stricter with fuzzy dominance this could be counteracted. Is fuzzy dominance a suitable method to select routes during CSA or will this cause useful routes to be lost?

Currently the slowest preprocessing step is calculating the possible walking distance with OpenTripPlanner (OTP) on walk only mode [25]. The high runtime is a result of the many one-to-one requests performed by OTP to calculate the walking distances, when this would be changed to many-to-many requests the runtime would be a lot lower. Every station would need one many-to-many request. Additionally the walking distance for Valys users in a wheelchair need to be precomputed, OTP has a feature for this.

At last the subsets of routes generated by the weighted sum and fuzzy dominance method in chapter 11 need to be compared with the subsets of routes real users would choose. When the new journey planner is in use for a while, a lot of data will be available about the routing preferences of the user. How can machine learning be used to show a better subset of routes?

It is important that as much data about the user and its routing preferences is stored as possible. Not only the journey the user books, but also the other journeys the user could choose from. This can then be linked to personal information about the user (age, gender, budget, aiding tools etc.). Also the restrictions the user puts on some of the criteria should be stored. All this information combined could be used in combination with machine learning to generate a more personalized subset of routes for the Valys user and also for other Valys users with similar characteristics. As a result of low data collecting of choices made by users, only a limited amount of relations could be found between characteristics of the trip or user with preferences in routing in part I.

REFERENCES

[1] 9292 | REISinformatiegroep BV / TeleAtlas, *Zonekaart Nederland,* https://9292.nl/gimmage/N2/DelfaultTemplate/Zonekaart_9292.pdf (2012), accessed 06-09-2017.

[2] T. D. Duin, R.P.W., *Prtools: A matlab toolbox for pattern recognition,* http://prtools.org/ (2005), accessed: 20-03-2017.

[3] AT&T Labs Research, *Graph Visualization Software (Graphviz),* http://www.graphviz.org/ (1991), accessed 25-04-2017.

[4] Google, *My maps,* https://www.google.com/maps/about/mymaps/ (2007), accessed 09-08-2017.

[5] J. S. LLC, *Mockplus,* https://www.mockplus.com/ (2014), accessed 02-08-2017.

[6] *Transvision,* https://www.transvision.nl (1994), accessed: 16-08-2017.

[7] M. van Infrastructuur en Milieu, *Koninklijk Nederlands Meteorologisch Instituut,* http://www.knmi.nl (1854), accessed: 20-02-2017.

[8] Centraal Bureau voor de Statistiek (CBS), *CBS-Gezondheidsenquête, CBS-GE,* (2014).

[9] E. W. Dijkstra, *A note on two problems in connexion with graphs,* Numer. Math. **1**, 269 (1959).

[10] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, *Contraction Hierachies: Faster and Simpler Hierachical Routing in Road Networks,* Proceedings of the 7th Workshop on Experimental Algorithms (WEA 2008) , 319 (2008).

[11] S. Funke, *Polynomial-time Construction of Contraction Hierarchies for Multi-criteria Objectives,* Alenex 2013 , 41 (2013).

[12] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, *Customizable route planning,* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) , 376 (2011).

[13] Y. Disser, M. Müller-Hannemann, and M. Schnee, *Multi-criteria shortest paths in time-dependent train networks,* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) , 347 (2008).

[14] D. Delling, T. Pajor, and R. F. Werneck, *Round-Based Public Transit Routing,* Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12) , 130 (2012).

[15] J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner, *Intriguingly simple and fast transit routing,* in *Experimental Algorithms: 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings*, edited by V. Bonifaci, C. Demetrescu, and

A. Marchetti-Spaccamela (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013) pp. 43–54.

[16] H. Bast, E. Carlsson, A. Eigenwillig, R. Geisberger, C. Harrelson, V. Raychev, and F. Viger, *Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns,* Algorithms – ESA 2010 , 290 (2010).

[17] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, *Route Planning in Transportation Networks,* Microsoft Research Technical Report , 1 (2015).

[18] H. Bast, M. Brodesser, and S. Storandt, *Result diversity for multi-modal route planning,* ATMOS-13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems-2013 **33**, 123 (2013).

[19] D. Delling, J. Dibbelt, T. Pajor, D. Wagner, and R. F. Werneck, *Computing multimodal journeys in practice,* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7933 LNCS**, 260 (2013).

[20] M. Farina and P. Amato, *A fuzzy definition of "optimality" for many-criteria optimization problems,* IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans. **34**, 315 (2004).

[21] T. Pajor, *Algorithm Engineering for Realistic Journey Planning in Transportation Networks,* (2013).

[22] *Google Maps Geocoding API,* https://developers.google.com/maps/documentation/geocoding/intro, accessed: 25-08-2017.

[23] OVapi, *Gtfs,* http://gtfs.ovapi.nl/ (2013), accessed 08-03-2017.

[24] Rijden de treinen weblog, *Afstandenmatrix mei 2017,* https://blog.rijdendetreinen.nl/2017/05/afstandenmatrix-mei-2017/ (2017), accessed 18-05-2017.

[25] OpenTripPlanner, *Opentripplanner,* (2009).

[26] Calendar42, *calendar42,* http://calendar42.com/ (2011), accessed 22-07-2017.

[27] PTALs, *Measuring Public Transport Accessibility Levels,* Chi 2007 (2010).

[28] K. MacCrimmon, *Decisionmaking Among Multiple-Attribute Alternatives: A Survey and Consolidated Approach,* (1968).

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II,* IEEE Transactions on Evolutionary Computation **6**, 182 (2002).

[30] P. Andre, M. L. Wilson, A. Owens, and D. A. Smith, *Journey Planning Based on User Needs,* Chi 2007 (2007).

[31] J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner, *Connection scan algorithm*, CoRR **abs/1703.05997** (2017).

Appendices

A

ADDITIONAL DATA ANALYSIS

The following figures (A.1, A.2 and A.3) give the total number of Valys trips per month, weekday or hour for 2016. December is the peak month of the year and the reason for this is that a lot of users visit their family for Christmas. Most trips occur during the weekend and this is probably due to children of the users not having to work during the weekend. When looking at the hour of departure it seems around 9/10 o'clock is the peak hour and there seems to be a peak just after every meal. Users do not seem to travel much during peak hours.

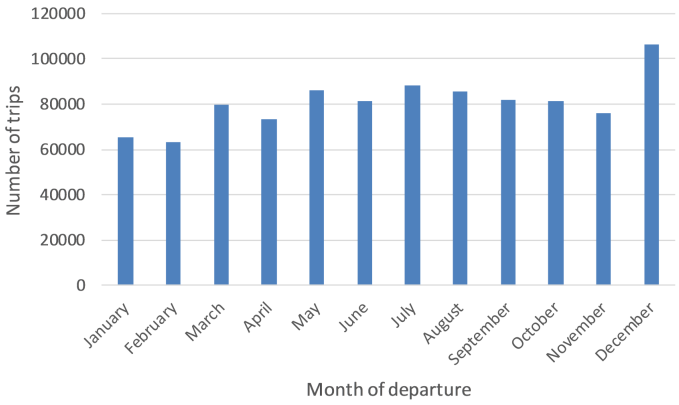


Figure A.1: Trip distribution per month of departure

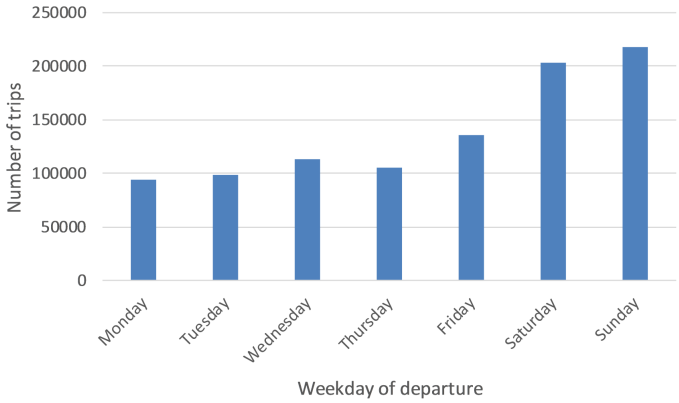


Figure A.2: Trip distribution per day of departure

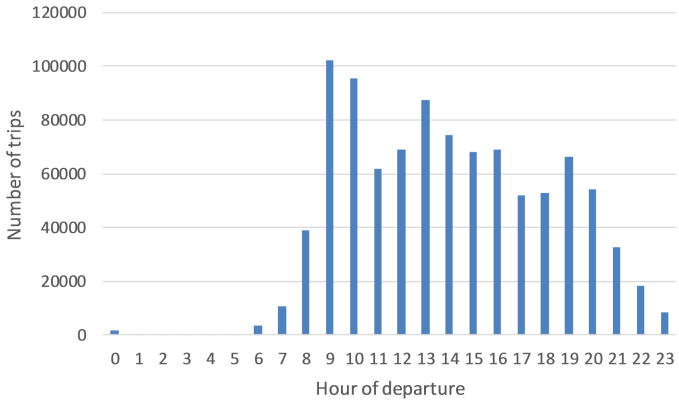
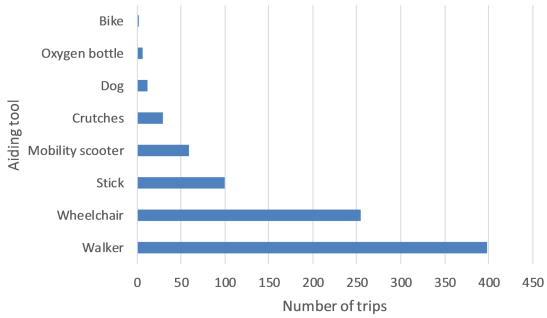


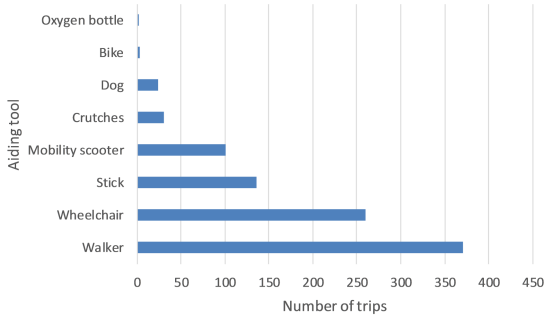
Figure A.3: Trip distribution per hour of departure

For 40% of the trips last year the user brought an aiding tool with them. The distribution of these aiding tools per Valys type is displayed in figure A.4a, A.4b and A.4c. The walker and the wheelchair are the most common used aiding tools for all types. For 29% of the trips the user had a client indication (e.g. visual impairment) and this is displayed per Valys type in figure A.5a, A.5b and A.5c. Visual impairment and impaired hearing are the most common client indications. The high PKB users are only a big part of Valys Basis. At the beginning of the year the *Persoonlijk Kilometer Budget* (PKB) of the Valys user is set again to its initial value. For most users this is 600 km and for users with a high PKB this is 2250 km. One can qualify to get a high PKB budget when he or she cannot travel by train due to a medical condition or other restriction. The user only loses kilometers

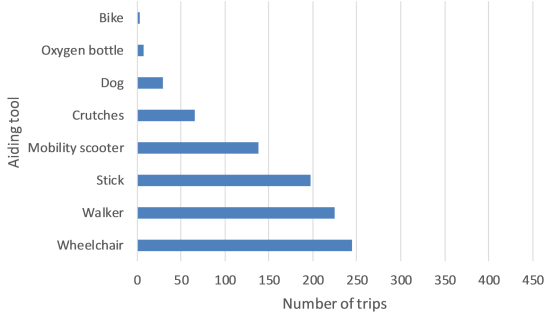
from his or her budget when using the taxi. It is interesting to note that (needing) a low entry is a bigger part of Valys Vrij than Valys Begeleid. The reason for this is that besides needing the low entry itself not much assistance is needed.



(a) Valys Basis

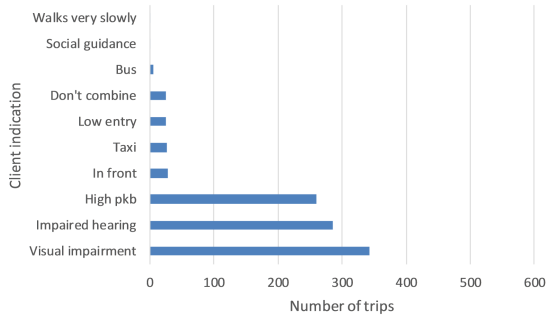


(b) Valys Begeleid

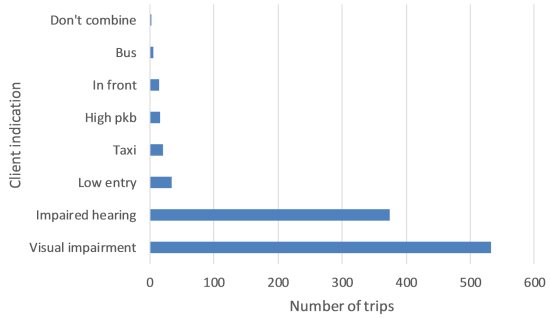


(c) Valys Vrij

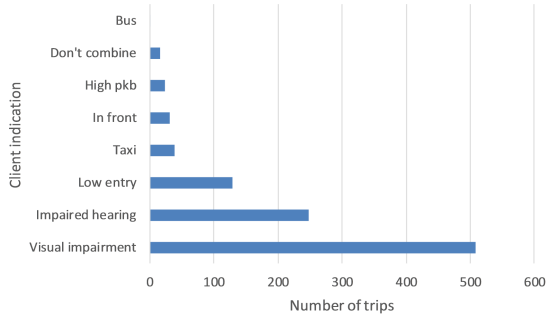
Figure A.4: Aiding tool distribution per travel type. The total number of trips is normalized to 1000.



(a) Valys Basis



(b) Valys Begeleid



(c) Valys Vrij

Figure A.5: Client indication distribution per travel type. The total number of trips is normalized to 1000.

B

ADDITIONAL MACHINE LEARNING

The chapter filters the trips and tries to find out which criteria of the trip or user can predict if someone travels partly by train or not.

B.1. TRIPS BETWEEN 30-90 KM

To limit the power of the total distance criterion, the trips are filtered on those with a total distance between 30 and 90 km. Hopefully this will increase the influence of other new criteria. Training and testing is done exactly as in section 3.3 and the average classification results per classifier are listed in table B.1 and table B.2 displays the (lowest) classification error when only one variable is used.

The lowest classification error (0.18) is a bit higher than in section B.2. When looking at the individual performance of the criteria it seems the power of the total distance variable has diminished a bit and the three other criteria seem to be more influential as they have a lower individual classification error. Other criteria do not seem influential enough.

Table B.1: Average classification error per classifier for trips with a total distance between 30 and 90 when training on train use and no train use. Equal number of trips for train use and no train use.

	No feature extraction	Fisher mapping	Principal Component Analysis (d=22)
nmc	0.32	0.21	0.31
ldc	0.21	0.21	0.22
fisherc	0.21	0.21	0.22
loglc	0.18	0.21	0.19
qdc	0.22	0.21	0.22
parzenc	0.3	0.21	0.3
bpxnc	0.2	0.21	0.2
lmnc	0.28	0.21	0.26
treec (p=15)	0.2	0.22	0.29

Table B.2: Classification error for individual criteria, when training on train use and no train use, that have an error below 0.4 for trips with a total distance between 30 and 90

Criteria	Classification error
Number of train trips in 2015	0.25 (treec/loglc)
Total distance	0.35 (treec/loglc)
Budget at the end of 2015	0.37 (treec/loglc)
Age	0.36 (treec)

B.2. ALTERNATING USERS

The trips are filtered on trips where the users participated in both trips partly by train and not partly by train in 2016. Is there a decisive factor that makes the user alternate between train use and no train use? To answer this question training and testing is done just as in the previous section. The average classification results are listed in table B.3. Besides the total distance there are no other criteria that have on its own a classification error below 0.4. Given that the lowest classification error is 0.24 one would assume there are other unknown criteria involved that makes the user alternate between travelling partly by train and not partly by train, otherwise the classification error would have been much lower. Possible criteria could be the mood of the user and if the user has done the same trip before by train.

Table B.3: Average classification error per classifier when training on train use and no train use for alternating users. Equal number of trips for train use and no train use.

	No feature extraction	Fisher mapping	Principal Component Analysis (d=22)
nmc	0.36	0.26	0.37
ldc	0.26	0.26	0.26
fisherc	0.26	0.26	0.26
loglc	0.25	0.25	0.25
qdc	0.39	0.26	0.29
parzenc	0.34	0.24	0.35
bpxnc	0.26	0.24	0.25
lmnc	0.26	0.25	0.26
treec (p=15)	0.24	0.24	0.31

C

CUSTOMER PANEL

A customer panel was held to find out from Valys users what can be changed about the route planning to improve their (train) journey or what can be changed about the route planning to convince non-train users to travel partly by train. In this context train users are users that used Valys Begeleid or Valys Vrij in the past year and non-train users are users that did not use Valys Begeleid and Valys Vrij in the past year. The customer panel distinguishes four different groups: train users without aiding tool, train users with aiding tool, non-train users without aiding tool and non-train users with aiding tool.

C.1. TRAIN USERS

Most users present at the customer panel indicated that 600 km was too low for their needs. The main incentive for users to travel partly by train is to be able to make more trips for the same amount of money, since users only spend a part of their budget when travelling by taxi and do not spend their budget when travelling by train. When a user completely spends his or her budget the price per km for the taxi increases from 0.20 to 1.31. How much budget the trips costs and the real cost itself are important factors for most users.

Another reason mentioned to travel partly by train is because it is not easy to get to a destination by only public transport. Some do not even have a train station nearby. One user indicated to still be able to drive, but that the walking distance from the parking lot to the destination was a problem and that Valys was the solution to limit the walking distance. Unexpectedly some even found the train more comfortable than the taxi, because there is more space in the train and others liked the social aspect of the train.

One user indicated to like the possibility to travel half by Valys Vrij. This specific user would like to travel by metro to the train station and from the next train station to the end destination by taxi. Valys Begeleid suits some users better than others. It is a good solution for someone who does not like to travel by him or herself, but other users may not like to obligated interaction that comes with it.

In the current system it is possible that a Valys Vrij trip takes longer than a Valys

Begeleid trip and this is due to that only the three nearest train stations from the location of departure and the three nearest train stations to the location of arrival are taken into account for the taxi to drive to or from when planning the journey. The assistance of Valys Begeleid is only at 31 stations in the Netherlands.

A man of 62 who was present at the customer panel represented a case where the Valys Begeleid trip was one hour faster than the Valys Vrij trip. In his case the Valys Begeleid version was in total 3 hours which consisted of a taxi to Station Utrecht, a direct train from Station Utrecht to Station Deventer and a taxi from Station Deventer to the arruval location. The two taxis combined would spend 54 km of the budget. In the Valys Vrij version of his case the travel time was 4 hours which included two taxis and four different trains of which only 14 km of the budget would be spend on the taxi rides. In this case the Valys Vrij version is best for someone who only wants to optimize the budget, but the Valys Begeleid version is best for someone who wants to optimize the travel time and/or the number of train transfers and this probably accounts for most Valys users. Both journeys should be possible for Valys Vrij.

Someone's age or aiding tool does not seem to give a direct preference in routing. It seems to be a personal matter if someone prefers for example a train ride with one transfer or a train ride with no transfers which is a half hour longer. The user should be able to decide which one is better in his or her situation. Transferring is considered to be exciting and stressful for some users, but cosy for others. Some only want to transfer when the train platform is the one across from the arriving platform. The assistance of Valys Begeleid is not always required for the complete journey, since some do not require this at train stations they come more often.

The needed average time of the questioned users to get from the taxi to the train platform ranges between 3 to 15 minutes, indicating the user should be able to adjust this to his or her own preference. The users gave a preference to not travel partly by train when they had a lot of luggage, were to travel a short distance (does not cost much budget), were to travel late at night or more comfort was wanted for a change.

Other improvements about the route planning include: being able to plan in toilet break time at a train station, being able to plan the train route itself (Valys then books the taxi rides around this) and being able to give a flexible return time. The users indicated to see no use in planning the route around avoiding rain.

Other possible improvements or comments for Valys or NS not related to route planning include:

- The NS discount card that provides a 40% discount after nine o'clock makes it impossible to arrive at the platform before nine when the station has a gateway. This can be a problem for the mostly old target group, especially during bad weather conditions.
- When someone uses the Valys card when travelling by train it is not possible to use NS discounts. Valys should allow discounts on their own card. Also travelling first class is not possible with their card.
- Increasing the amount of Valys Begeleid stations.

C.2. NON-TRAIN USERS

Also most non-train users present at the customer panel found a PKB of 600 km too low for their needs. The group without aiding tool gave different reasons for not using the public transport:

- They are afraid to get robbed / afraid for misbehaving youth
- They are afraid there is not place to sit
- They used to go with their spouse, but he or she passed away
- They do not want to transfer
- They are not sure what to do when something about the public transport changes (delay, cancellation)
- They find the travel time too long when travelling partly by train.
- They do not have a train station nearby
- They have a difficulty with their OV chipcard
- They have had a bad experience where the train was delayed a lot or cancelled
- They would need Valys Begeleid, but they do not want to be patronized.

The non-train users without aiding tool seem to be a difficult group to convince to travel partly by train even after watching a video about how Valys Begeleid works. Most of them blame it on their condition, for example one of them had a balance disorder. Of the reasons mentioned above only not transferring and cancellation or (much) delay of the train can be solved. This last reasons is mostly solved already by Valys by bringing the user immediately to the end destination

The user group with aiding tools seem to have a different outlook on the matter of travelling by public transport. Daily life itself is already a challenge for them and adding travelling by public transport is not much more of a challenge. Even though this group did not travel with Valys Begeleid or Valys Vrij last year some still used the public transport last year. The present blind people of this group would like to only transfer at train stations with an information desk in case they had a question about the train station and they suggested someone should create an app that can give direction (in sound) on the train station to the right platform. In case the guide dog of the blind person is not allowed at the end destination, Valys Begeleid could be a solution.

Just like the train users from the previous section they have a preference to be able to decide their own transfer time, walking speed and maximum number of transfers for the route.

C.3. ROUTE PLANNING

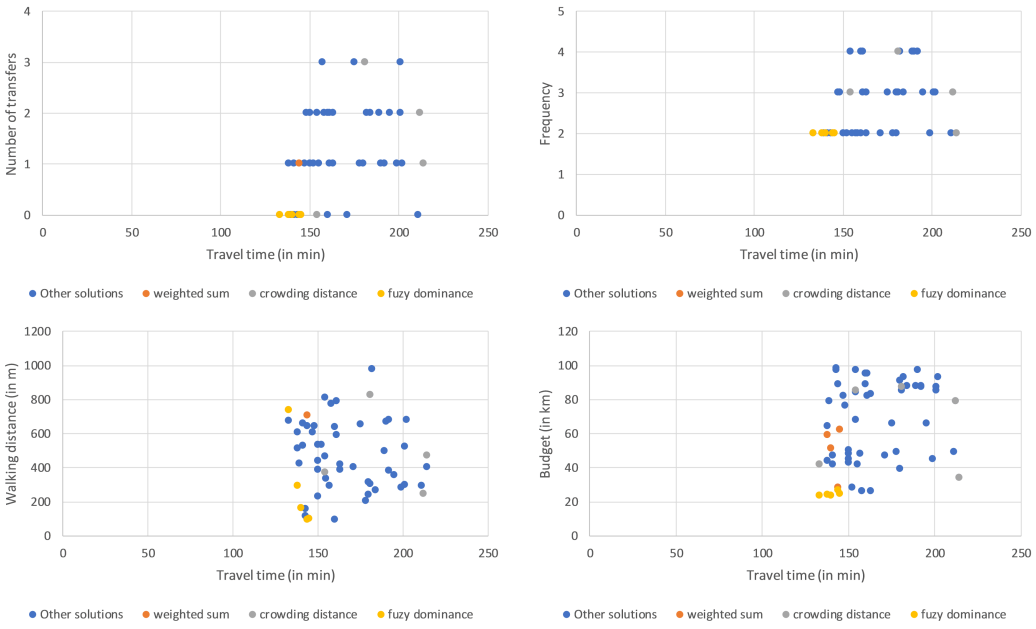
From the customer panel multiple things about the route planning can be concluded. The current algorithm can give longer travel times for Valys Vrij trips than Valys Begeleid trips when only small stations are nearby. The users would like to have a more personalized journey planner. Someone's age or aiding tool does not seem to give a direct preference in routing, it seems to be more of a personal matter.

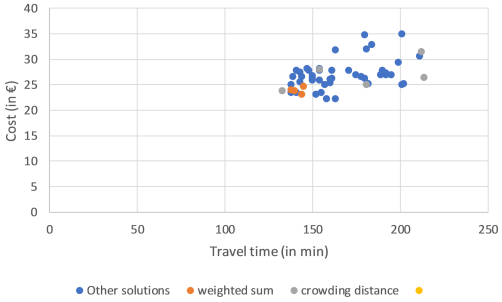
Besides the basic optimizations (travel time and numbers of transfers), the following optimizations are interesting for this target group: walking distance, cost and budget. This could lead to different arrival and departure train stations based on the given preference. In case the train nearby does not ride often in the desired direction optimizing on frequency could also be useful. Mentioned restrictions were maximum walking distance, walking speed, maximum number of transfers (including no transfers), only transfer when the platform is the one across from the arriving platform and only transfer at stations with an information desk.

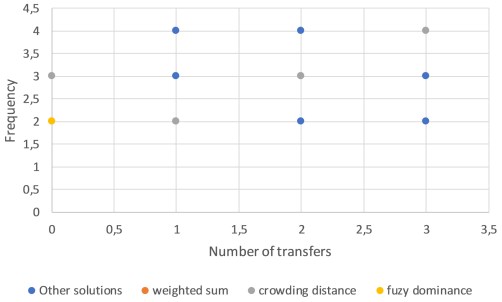
D

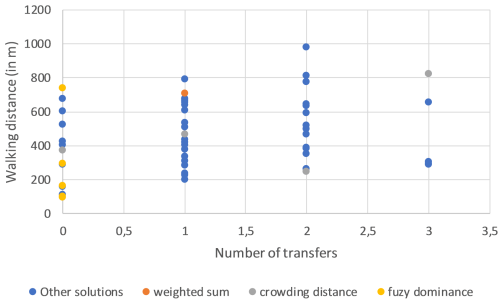
Visualization Pareto-optimal routes

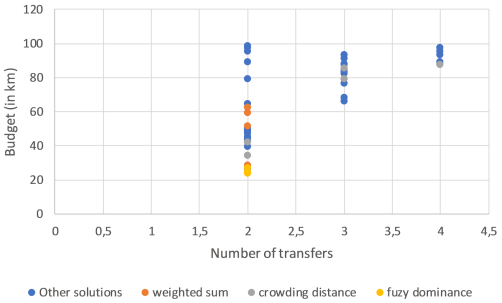
Figure D.1 visualizes the chosen journeys from the Pareto-optimal set of journeys for the example Valys trip (chapter 11). This is done for the weighted sum, crowding distance and the fuzzy dominance method. The values for the criteria of the journeys for the weighted sum and the fuzzy dominance method are closer to the origin than those for the crowding distance method.

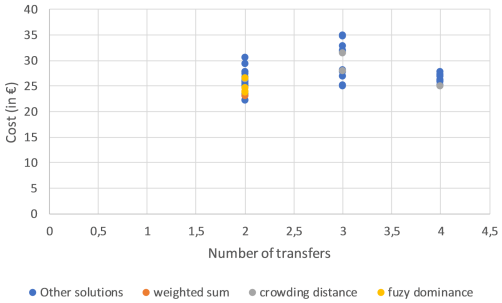


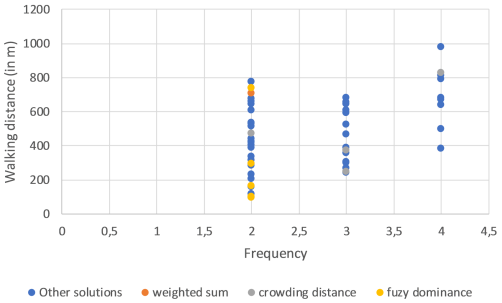

Cost (in €)
Travel time (in min)
Other solutions
weighted sum
crowding distance

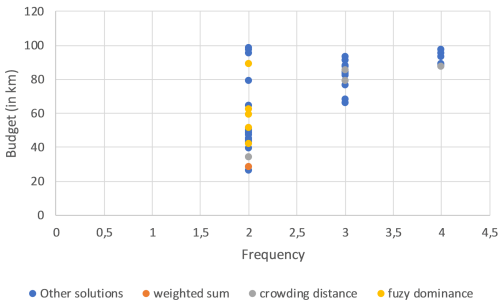

Frequency
Number of transfers
Other solutions
weighted sum
crowding distance
fuzy dominance

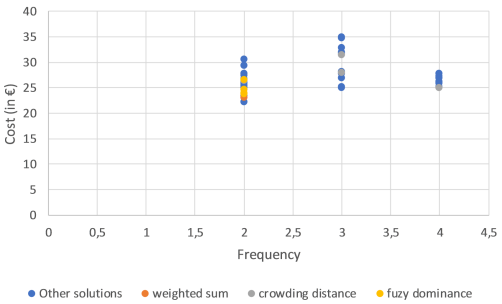

Walking distance (in m)
Number of transfers
Other solutions
weighted sum
crowding distance
fuzy dominance


Budget (in km)
Number of transfers
Other solutions
weighted sum
crowding distance
fuzy dominance


Cost (in €)
Number of transfers
Other solutions
weighted sum
crowding distance
fuzy dominance


Walking distance (in m)
Frequency
Other solutions
weighted sum
crowding distance
fuzy dominance


Budget (in km)
Frequency
Other solutions
weighted sum
crowding distance
fuzy dominance


Cost (in €)
Frequency
Other solutions
weighted sum
crowding distance
fuzy dominance

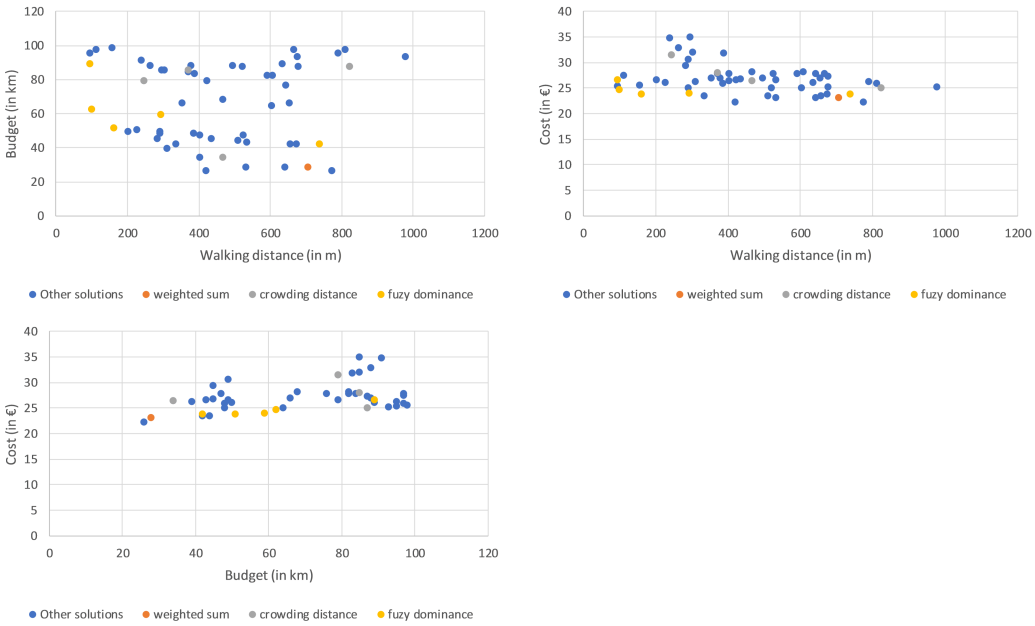


Figure D.1: Subset Pareto-optimal routes visualized for the weighted sum, crowding distance and fuzzy dominance method.