Calibrating OD-Matrices with Public Transport and Mobile Phone data

MSc thesis, Ben Rorije
Deventer/Enschede
June 16, 2011

UNIVERSITY OF TWENTE.

OmniTRANS
Transport Planning Software

Documentation Page

Title	Calibrating OD-Matrices with Public Transport and Mobile Phone data Master thesis
Author	B. Rorije University of Twente Discrete Mathematics and Mathematical Programming benrorije@googlemail.com
Supervisors	Dr. G.J. Still University of Twente Discrete Mathematics and Mathematical Programming ir. M.P. Schilpzand Omnitrans International Deventer
Keywords	OD matrix, calibration, public transport, mobile phones, OV-Chipkaart, GSM, GPS.
Date of publication	June 16, 2011

Summary

In traffic modelling the displacements of people between different zones are modelled. These displacements are stored in an Origin-Destination matrix (OD matrix). These matrices are often based on survey data or on a mathematical model. To improve these matrices so that they approach reality they are calibrated with real life data. At the moment this is mainly done with traffic counts.

In this thesis two different sources of information are studied, to see in which way they can also be used for OD matrix calibration. These sources are information from the public transport, and information from mobile phones. For public transport information three different models were defined, based on either counts, stops and journeys. For mobile phone datasets an algorithm was design to convert the data to a route in the route network. These four models were then implemented in the OmniTRANS program, which resulted in some conclusions and recommendations.

Samenvatting

In verkeersmodelleringen worden de verplaatsingen van mensen tussen verschillende zones gemodelleerd. Deze verplaatsingen worden opgeslagen in een Herkomst-Bestemmings matrix (HD matrix). Deze matrices zijn vaak gebaseerd op een enquête of op een wiskundig model. Om deze matrices te verbeteren zodat ze beter de werkelijkheid benaderen worden ze gekalibreerd met actuele data. Op het moment is dit vooral gedaan met verkeerstellingen

In deze thesis worden twee verschillende bronnen van informatie bestudeerd, om te zien hoe deze kunnen worden gebruikt voor HB matrix calibratie. Deze bronnen zijn informatie uit het openbaar vervoer, en informatie van mobiele telefoons. Voor de openbaar vervoer informatie zijn drie modellen waren gedefinieerd, gebaseerd op tellingen, haltes en ritten. Voor mobiele telefoon datasets is een algoritme ontworpen om de data om te zetten naar een route in het wegennetwerk. Deze vier modellen zijn daarna geïmplementeerd in het OmniTRANS programma, wat in een aantal conclusies en aanbevelingen resulteerde.

Preface

This thesis is the final part of my masters study Applied Mathematics at the University of Twente. It is part of the chair of Discrete Mathematics and Mathematical Programming, and the practical part of the thesis was done at Omnitrans International in Deventer, the Netherlands.

With this thesis the end of my masters study has been reached, and I look back on a fun and interesting time here in Enschede. Aside from studying I have been active at several cultural societies, which taught me a whole lot.

For this thesis I would like to thank my two supervisors, Maarten Schilpzand and Georg Still. They helped me a lot during the course of this thesis, and gave good advice on the structure and contents of this report. I would also like to thank my colleagues at Omnitrans International. It was fun working there, and there was always time to answer my questions.

Ben Rorije
Deventer / Enschede, June 2011

Contents

Notations

An overview of all variables and symbols as they are used in this report.

Standard variables in a Network:

C	set of all centroids, $\lvert C\rvert = c$
	with centroids $i, j \in C$
N	set of all nodes, $\lvert N\rvert = n$
	with nodes $p, q \in N$
A	set of all links, $\lvert A\rvert = a$
	with link $l \in A$
S	set of all stops, $\lvert S\rvert = s$
	with stops $f, k \in S$
T	set of all (public) transit lines, $\lvert T\rvert = t$
	with transit line $o \in T$

Variables used for Calibration:

$\mathbf{g}$	OD matrix
	with OD value $g_{ij} \in \mathbf{g}$
$\hat{g}$	starting matrix for OD matrix calibrations
I	set of all OD pairs, $\{(i,j) \mid i, j \in C,\ i \neq j\}$
R	set of all restrictions
	restriction $r \in R$
$\hat{A}$	set of links for which a traffic count is known
$\mathbf{v}$	set of all flows
	flow $v_a \in \mathbf{v}$ on link $a \in A$
$\hat{\mathbf{v}}$	set of all flows belonging to a count
	flow $\hat{v}_a \in \mathbf{v}$ on link $a \in \hat{A}$
P_{ijr}	Load on the path between centroids i and j that is used for restriction r
ϵ_r	Elasticity of restriction r
S_l	Screenline matrix belonging to link l
χ_0	the initial calibration factor

Variables used for Public Transport:

H	is the matrix containing all OV-Chipkaart data; $H(f,k)$ is the number of people going from stop f to stop k
N_f^{before}	is the set of stops coming before stop f, on any transit line
N_f^{after}	is the set of stops coming after stop f, on any transit line
$paths(i,j)$	is the collection of all transit paths between centroids i and j
P_{ijp}	is the percentage of flow going over path $p \in paths(i,j)$
$TL^p(l)$	is the transit line attached to link l, on path p

Variables used for Mobile Phones:

N_i	all nodes inside area i
G(V,E)	Graph describing the network, (with V the set of nodes and E the set of links)
AB	The link between nodes A and B
$d_e(A,B)$	The Euclidean distance between two points A and B
Q_i	A GPS point, consisting of a pair of coordinates (x_i, y_i)
$Q_{1..T}$	Set of points given by the GPS data
Q^t	The projection of Q on the link AB
$p\{E_1..E_p\}$	The path along links E_1 to E_p (with E_i, E_{i+1} sequent)
$Phones_{\text{ on,vehicle}}$	the number of active phones in a car
$NofOccupants$	the number of people in a car
$Pct_{\text{phones/person}}$	the percentage of people that have a mobile phone
$Pct_{\text{phones on}}$	the percentage of mobile phones actually turned on
NC_k	the real number of cars traversing route k
NP_k	the number of phones traversing route k
R_k	the set of traffic counts attached to route k
C_r	the total value of count r
ϵ_r	the elasticity of count r

1. Introduction

Before starting any project it is important to look at what the goal of the project is, and what will have to be done to reach that goal. In section 1.1 a small background of the problem will be given, in section 1.2 the main project will be discussed, with the two steps that will be taken. And in section 1.3 an overview will be given of the structure of this thesis.

1.1 Background

In the area of traffic modelling and planning the OD matrix takes an important place. An OD matrix shows how many displacements there are from one location to another, which is a vital part of any traffic or transportation model. Because OD matrices are so much in use it is also important to have matrices that are 'up to date', meaning that they should give an accurate representation of the actual world. Keeping OD matrices up to date is done by performing calibrations, altering the OD matrices with information from the actual world.

In current OD matrix calibrations most information is taken from ordinary traffic counts. This works fine if they are applied to an older version of the OD matrix, or a matrix generated by a mathematical model. But sometimes traffic counts just aren't enough and extra information is needed. In recent years there have been two developments that might give this extra information.

1.1.1 Public Transport

Aside from vehicles there is a large area of public transport journeys that is also included in many models. But the OD matrix for public transport is often left uncalibrated, either because public transport isn't deemed important enough, or because only a small amount of information is available. But with the introduction of a Dutch electronic travel card, the 'OV-Chipkaart', that might change. Information from the OV-Chipkaart will give a sea of information about passengers in the public transport, which is exactly what is needed for the calibration of OD matrices.

1.1.2 Mobile Phones

In recent years the number of mobile phones has increased dramatically. Many people today are carrying a mobile phone with them at all times, to stay in touch with others. When a mobile phone is switched on, it is tracked by the mobile network, in case someone else tries to call that person. But that also means that the location of anyone with a mobile phone is (in theory) known. And the location of a person can, amongst other things, be used for OD matrix calibration.

1.1.3 Modelling

Both the data from the public transport and from mobile phones is more than just counts on a link. They both have information about (a part of) a whole route from one person. If the data from both sources is just converted to counts this information will be lost. Therefore part of the modelling process will consist of trying to include as much information as possible into the calibration.

With the data from mobile phones there is another aspect. Mobile phone data shows only a percentage of all displacements, as not everyone will have a mobile phone. Therefore it must be studied if perhaps the data is representative to the rest of the population. In which case the data can perhaps just be multiplied to get the full number of displacements, which would save a lot of work.

1.2 Main Project

The main subject of this thesis is to research if information from public transport and from mobile phones can be used for OD matrix calibration and, if so, in which way. This research can be divided into two steps: Modelling and Implementation. In both phases the subjects of public transport and mobile phones will be treated separately, though some research has to be done to see if there is perhaps some overlap between the two.

1.2.1 Modelling

In the modelling phase the existing model for OD matrix calibration has to be extended. The information of public transport and mobile phones probably has a different structure than ordinary counts, so it will also have a different part in the model. The information has to be studied, to see what is possible and what adaptations are practical, also from the viewpoint of (a possible) implementation.

1.2.2 Implementation

After the modelling phase is complete it is also important to check if the proposed models can be implemented and do in fact work. Therefore a proof-of-concept has to be build, a prototype of the proposed adjustments. Depending on the format of the data, it has to be converted before it can be used. After that the implementation should work on several test-cases, proving the added functionality and better results.

1.3 Structure of the Thesis

This thesis is build up into four parts. The first part, Part 0 consists of the literature research done at the start of the final project. In Chapter 2 the general part of the literature research is given. In Chapter 3 an introduction is given into the company for which this thesis was written, Omnitrans International, and their program, OmniTRANS. And in Chapter 4 some specification is given to the above problem description, based on all the information from the literature research.

After the literature research the two areas, public transport and mobile phones, are treated separately. In Part I the public transport is treated. For this area first some introduction is given in Chapter 5, together with the first steps into the modelling. Then in Chapter 6 some algorithms are defined and discussed, to help with the modelling of the public transport model. In Chapter 7 then three options are treated, that flow from the model and the different algorithms. Then in Chapter 8 the implementation of these three options is given, with a small overview of the encountered problems and their solutions. And finally in Chapter 9 some tests are discussed, together with the results gathered from these tests.

In Part II the modelling and implementation of the mobile phone area is discussed. In Chapter 10 first a small introduction is given, together with an outline of the needed steps to successfully model the mobile phone datasets. In Chapter 11 two algorithms are given to convert the mobile phone datasets into routes through the network. Then in Chapter 12 some steps are given into processing the routes, such that they can be used for OD matrix calibration. In Chapter 13 the implementation of one of the two algorithms is given, together with some adaptations made to the algorithm. And in Chapter 14 again some tests are done on the algorithm, and from these tests some results are given.

In the final part, Part III, the final remarks and conclusions of this thesis are given. First in Chapter 15 some mathematical remarks are given, that were already referenced during the course of this thesis. In Chapter 16 three more general remarks are given, also referenced earlier in this thesis. And finally in Chapter 17 the final conclusions of this thesis are given, together with some recommendations for the future.

2. Literature Review

In this chapter the results of the Literature Review will be given. In section 2.1 some general background information is provided as an introduction into the research area. In section 2.2 the calibration of OD matrices is treated, together with some solution methods. Then in section 2.3 an overview of public transport and the OV-Chipkaart is given, while section 2.4 contains a background of mobile phone data.

2.1 General Background

Transport Modelling is a very large field with a lot of different topics in it. Most models in transport modelling have to do with networks and the displacements in them, people or goods moving from one place to the other. This report will mainly focus on traffic modelling, the modelling of vehicles and persons in cities and/or other areas. And then especially on improving an existing model, by calibrating certain parts of the model.

2.1.1 Four-step Model for predicting traffic flows

In the area of traffic modelling the following Four-step model is often used [Ortúzar & Willumsen, 2001]. It has been around for more than fifty years, and lies at the basis of most transportation models used today.

The Four-step model assumes that a network of links is already known. First the area is divided into different zones, when the area is a city these zones are often modelled on the postal codes. All zones are connected to each other by the network of links (roads/tracks/other). These zones are then filled with socio-economic data, consisting of the number of residents in that zone, the number of jobs in that zone and any other attractions that are located in that zone (i.e. shopping centres, amusement parks, etc.). After adding the socio-economic data the following four steps are applied.

1. **Trip generation**

   In the first step the socio-economic information from each zone is translated into trip data: the number of trips (each trip is one person/car/transport/other) that are leaving from a zone and the number of trips arriving at a zone. This information quantifies the 'production' (departures) and the 'attraction' (arrivals) of a particularly zone. All trips are tagged with a purpose, which can be something like 'work' or 'personal'.

2. Trip distribution

After step 1 only departures and arrivals are known. In the second step these are connected, as a departure from one zone belongs to an arrival at another zone. These trips are then stored in an Origin-Destination matrix (OD matrix), defining the number of trips between each origin and destination. Linking the origins and destinations can be preformed by several different methods. Two methods often used are the Fratar model (also called Growth Factor model) and the Gravity model. The Gravity model will be treated in the next section.

3. Modal split

With all the trips known, it becomes important to look at the mode of each trip, i.e. what kind of transportation is used. The mode can be by car or public transport (trains, trams, buses, etc.), but other modes can also be used (for example cycling or walking). For every mode a separate OD matrix is created, only containing trips belonging to that specific mode.

4. Route Assignment

Now that every trip has an origin, a destination and a mode the exact route of the trip can be assigned. Route assignment is the most difficult step of the problem, as there are many things that have to be taken into account. For example some links have a maximum load, other links can only be used for one type of mode (highways for cars, train tracks for trains). Another difficult aspect is congestion: the route of a trip is dependent on congestion, but also influences it. This is why often an iterative approach is used, searching for a user equilibrium.

2.1.2 Gravity Model

The gravity model is based on the principles of gravity. Gravity says that two objects always attract each other, and that this attraction is based on the mass of the two objects, m_1 and m_2, and the squared distance between them, r^2. In formula the above can be written as:

$$F = \frac{Gm_1m_2}{r^2}$$

For transport modelling the formula becomes:

$$A_{i,j} = \frac{\alpha P_i P_j}{(d_{i,j})^2}$$

In the formula the trip goes from origin i to destination j, with $A_{i,j}$ the attraction between these two. P_i and P_j are the population sizes of zones i and j, respectively, and $d_{i,j}$ is the distance between zones i and j.

The above formula is of course a very simple model, and not very accurate. An improvement is to look at the number of trips arriving and leaving a zone, not the population. The measure of distance is also a bit vague, as in a road network some roads can be traversed quicker than others, meaning that distance is perhaps not the best measure to use. Together this leads to:

$$A_{i,j} = \alpha O_i D_j f(i,j)$$

with O_i the number of departures from origin i, D_j all arrivals at destination j, and $f(i,j)$ a general measure of the travel cost between zones i and j ([Ortúzar & Willumsen, 2001]).

The implementation of the Gravity model into OmniTRANS can be found in section 3.4.

2.2 OD Matrix Calibration

Between steps 3 and 4 in the Four-step model another step is needed. Because the OD matrices from steps 2 and 3 are not based on 'real' results (they are mathematical models), they are not guaranteed to have good results. Therefore the OD matrices have to be calibrated to approximate the real world. There are several different methods that can be used to calibrate an OD matrix.

In the past OD matrices were calibrated with information that came from a whole range of surveys, everything from roadside interviews to home interviews and license plate tracking. But these methods are very expensive and time consuming, as a lot of information is needed to create a good OD matrix. Therefore researchers looked at other, more easily obtainable data, to use for creating a good OD matrix. Eventually they decided on using traffic counts. Traffic counts are easily obtainable, and are used in a wide variety of applications, which means that they are also widely available.

Note: In the literature sometimes the term "OD matrix estimation" appears. This term is more used to indicate that the OD matrix is created from the counts or surveys, and not changed (calibrated) by counts or surveys. Therefore the term OD matrix estimation really means something else, and is not applicable for this thesis.

2.2.1 Problem specification

Every origin and destination is located on a road network. A road network is build up from a set of nodes, called N, and the set of links between these nodes, called A. Traffic counts are known for some links, the set of these links is called $\hat{A}$. On every link there is a flow of traffic, the set of all flows is called $\mathbf{v} = \{v_a |\ a \in A\}$. For the links in $\hat{A}$ the flow is known from traffic counts, the set of these flows is called $\hat{\mathbf{v}} = \{\hat{v}_a |\ a \in \hat{A}\}$.

Origin-Destination matrices contain all the possible pairings between an origin and a destination. Define the OD matrix to be $\mathbf{g}$, then the number of people going from origin i to destination j will be g_{ij}. Intrazonal trips are not taken into account, therefore $g_{ii} = 0$. Calibrating an OD matrix works with a 'starting' matrix, $\hat{\mathbf{g}}$, derived from the Four-step model. The starting matrix will be calibrated with the counts from $\hat{A}$ to create the 'target' OD matrix $\mathbf{g}$.

This gives rise to the following General Problem formulation (GP):

$$\begin{aligned} \min_{\mathbf{g},\mathbf{v}} F(\mathbf{g},\mathbf{v}) &= \alpha F_1(\mathbf{g},\hat{\mathbf{g}}) + (1-\alpha)F_2(\mathbf{v},\hat{\mathbf{v}}) \\ s.t.\ \ \mathbf{v} &= assign(\mathbf{g}), \\ \mathbf{g} &\geqslant 0, \\ \mathbf{v} &\geqslant 0. \end{aligned} \tag{GP}$$

In the above description F_1 and F_2 are the distance measures between $\mathbf{g}$ and $\hat{\mathbf{g}}$, and between $\mathbf{v}$ and $\hat{\mathbf{v}}$, respectively. α is a constant on the interval $[0,1]$, and indicates the level of confidence in the original matrix. If α is close to 1 the new OD matrix lies close to the old OD matrix, if α is close to 0 the new OD matrix is heavily influenced by the traffic counts.

The assignment of $\mathbf{g}$, represented by $\mathbf{v}$, is on its own also an optimization problem. $\mathbf{v}$ is the solution of a route assignment, assigning the OD flows from $\mathbf{g}$ to routes on the network, subject to the constraints on the road network, and therefore $\mathbf{v}$ is the solution of a user equilibrium. This means that **(GP)** is a bi-level problem, in that there is interaction between the constraints and the objective function. In each iteration a new route assignment is made, changing the values $\mathbf{v}$ and therefore the influence of $\hat{\mathbf{v}}$.

There are many methods and algorithms for the route assignment of $\mathbf{g}$. In Chapter 3 two methods for a route assignment will be treated, as part of the introduction into Omni-TRANS.

2.2.2 Solving methods

For solving the general problem above there are several methods. These methods can be divided into three areas: Minimizing Information, Statistically Interference and Gradient Based Solutions. In the next section they will be briefly treated, using information from [Abrahamsson, 1998] and [Spiess, 1990].

- **Minimizing Information**
  The Minimizing Information approach works from the premise that the information from traffic counts will never uniquely define the target OD matrix. Therefore this method tries to minimize the amount of information that is added to the starting matrix in the calibration. That way the end result is an OD matrix, changed to include all the traffic counts, but still as close as possible to the starting matrix. The Minimizing Information method is sometimes also called the Entropy Maximizing approach, as minimizing information is the same as maximizing entropy.

- **Statistically Inference**
  The idea behind Statistical Inference approaches is that they see the starting matrix to be generated by a probability function. The differences between the target OD matrix and the starting matrix is just the result of variance. Therefore these methods try to estimate the parameters of the distribution they think was used, and with those parameters reconstruct the target OD matrix.

There are three main methods in the area of Statistically Inference:
- Maximum Likelihood,
- Generalized Least Squares,
- Bayesian Inference.

These three differ in the distribution that they use, and therefore also in their definition of the objective functions F_1 and F_2.

- **Gradient Based Solutions**
  With Gradient Based solutions the starting OD matrix is changed, in the 'direction' based on the gradient of the objective function. Every time a traffic count is added the objective function changes, and the algorithm will look in a new direction. Because the gradient is used for determining the optimal direction, this direction will always point to the largest yield.

  The Gradient method is probably the most widely used algorithm today, and therefore it will be used in the rest of this report, together with OmniTRANS' own algorithm. The specific Gradient algorithm that will be used will be the one from [Smits, 2010], which will be further explained in section 3.7.

2.3 Public Transport

Public Transport is the generic term for a lot of different types of transport. The fact that it is called 'public' comes from the fact that companies providing public transport are (partly) funded by the government. This fact also means that they are required to offer their services to everyone. Most public transport services are bound by a (strict) timetable.

2.3.1 PT Modes

As said before, there are a lot of different types of public transport. Public transport can be categorized in the following three 'modes' [Wikipedia, 2010]:

- **Buses and Coaches**

  Buses and coaches are large car-like vehicles, operating on conventional roads. Both transport passengers from and to stops, which are often specially marked and placed along their route. They have a low capacity in comparison to track-based transport, and therefore are used mostly for short distances or for routes where there are no track-based alternatives.

- **Track-based Transport**

  Track-based transport uses, as the name implies, tracks over which the transport moves. Most track-based vehicles are powered by electricity, either by overhead lines or through a third rail between the tracks, or are powered by diesel (on tracks without overhead lines, or in countries with a limited electric network).

Track-based transport comes in three types:

- *Trains*

  Trains transport passengers between cities, following a specific route along a series of cities, towns, villages and stations. Trains follow a strict timetable, as they are not influenced by normal congestion on the road network.

- *Trams and Light rail*

  Trams are track-based vehicles that drive (at least partially) on the conventional roads, mixing in with the rest of the traffic. Trams are smaller than trains, and therefore also have a lower capacity. Light rail is a combination of a tram and a train, designed in such a way that it can both ride on tram tracks and train tracks.

- *Rapid Transit*

  Rapid Transit is a generic name for both the Metro, Subway and the Underground. In general rapid transport is quicker than trams and buses, and has more capacity. The difference is that rapid transit is often based in the tunnels beneath a city, needing special entrances and exits.

• **Ferry**

  Ferries are boats or ships transporting passengers and vehicles over water. In the case of public transport ferries are often small boats, used for transporting cars and pedestrians across a (small) river or channel. Some ferries have strict timetables, leaving at specific times, others will only depart if enough people are on board.

2.3.2 OV-Chipkaart

In recent years a lot of countries have adapted some sort of electronic travel ticket or card for their public transport [Pelletier *et al.*, 2009]. The appeal of electronic travel cards is easy to see: it prohibits fare dodgers, gives companies insight in the travel habits of passengers and makes it easier for the passengers to pay for their tickets. This section will give an overview of the Dutch version of the electronic travel card, the “OV-Chipkaart”.

Background

(Most information in this section comes from [Ministry of Transport, Public Works and Water Management, 2008] and [OV-Chipkaart.nl, 2010])

The idea for a nation wide electronic travel card already existed in 1992, when the Nationale Spoorwegen (the Dutch Railways, NS) got funding to test electronic tickets in several towns. This test was a success and in 1993 plans for the implementation of the OV-Chipkaart were published. The actual introduction of the card was delayed several times, but in 1995 the first OV-Chipkaart could be used.

At the moment there are two regions (Amsterdam and Rotterdam) where the OV-Chipkaart is the only valid payment method in the metro, tram and bus. In the provinces Groningen and Drenthe, and in trains run by local companies, the OV-Chipkaart had some delays, and for the moment cannot be used at all. In the rest of the country and in the trains of the NS a dual system is in place: both the OV-Chipkaart and the old payment options are valid.

The OV-Chipkaart has three different card types, which are differentiated by who will use them.

- **Personal card**
  Personal cards are for people who travel a lot with the public transport, or who have a subscription to a specific route or discount (people over 65, students, etc.). They feature a photograph and some personal information about the holder, so it cannot be used by anyone else.

- **Anonymous card**
  Anonymous cards are for people who travel only once in a while, or people who share their card with others. They do not feature any personal information, and are therefore also used by people who prefer not to share their personal information with the travel companies.

- **Disposable card**
  Disposable cards are for people who rarely use the public transport, like tourists. After it has been used it can be disposed easily, as it cannot be recharged.

Not much is known about when the OV-Chipkaart will be the only valid payment method for the public transport in the Netherlands. The NS already reported that they will keep accepting paper tickets until the majority of passengers have switched to the OV-Chipkaart. There have also been some problem with transfers from one company to another: they sometimes resulted in the passenger paying too much. Minister of Transport, Public Works and Water Management Eurlings responded that the full implementation of the OV-Chipkaart will be delayed until that issue is resolved [Minister C. Eurlings, 2010].

Data from the OV-Chipkaart

One of the reasons the OV-Chipkaart was introduced was for the information it gave to the public transport companies ([Ministry of Transport, Public Works and Water Management, 2008]). Before they introduced the OV-Chipkaart, they had to hold numerous surveys and counts just to get an idea of how many people used which services on which time.

Now they can get this information easily from the data generated by the OV-Chipkaart. Public transport companies use this information to check the popularity of certain lines and services, so they can quickly change to bigger/smaller vehicles if needed.

In the area of OD matrix calibration there have been almost no publishings on how to use information from the public transport. Two studies that do use data from electronic travel cards, [Chan, 2007] and [Cui, 2006], only use it in a special model where only public transport is taken into account. No studies were found that actually used public transport data together with the regular traffic data in one model.

Information from electronic travel cards is however used for a wide variety of other studies. For example some companies are trying to find demographic groups for additional marketing. Other companies want to develop real-time information on conditions inside the network, in case additional personal or material is needed. There also have been several studies trying to find the precise path used by passengers (which is only useful in networks where multiple routes are possible) [Pelletier *et al.*, 2009].

2.4 Mobile Phones

Mobile phones are starting to become an integral part of our society. They give people the ability to call and be called, even when they are not at home or at work. They allow them to send short messages to each other, and even to surf the internet or make photographs. In recent years this has escalated with the introduction of so-called "smart phones", mobile phones that offer much more in the way of processing power, connectivity and applications.

Mobile phones use special mobile communication networks. These networks are managed by a mobile operator, for example Vodafone or Orange. How the different networks function is based on the kind of system that is used. The largest mobile communication system that is in use today is the Global System for Mobile communications (GSM) network, and therefore for this thesis only that network will be considered.

Another aspect of mobile phones today is that some of them are carrying a Global Position System (GPS) locator with them. A GPS locator interfaces with satellites in the sky, giving the user the coordinates of his or her location. These locations are sometimes stored, and can be used for finding the exact route of that mobile phone. Therefore also GPS datasets will be considered in this thesis.

2.4.1 Global System for Mobile Communications

As mentioned before the GSM system is the largest mobile phone system in use today. According to the representative of the GSM system, GSM Association, in 2009 80% of all mobile phones used the GSM network [GSM World - Market Data, 2010]. Therefore GSM datasets will give large datasets, which can be used for the OD matrix calibration.

Background

The GSM system differs on several points from other systems. For one they use a so-called SIM (Subscriber Identity Module) card. These cards contain a unique identification number that identifies a user on the network, so that the network knows which user can be reached at which transceiver. Because this information is located on a card it can be removed from one mobile phone and inserted into another, giving more freedom to users. Other systems do not have this ability, and therefore users cannot easily switch between mobile phones.

GSM is also the system that introduced the term roaming to mobile communications. When someone is using a mobile phone that phone has a connection to a nearby transceiver. But if the person is travelling the mobile phone might leave the range of that transceiver, and enter the range of the next one. The GSM system then automatically hands the connection over to the next transceiver, allowing the mobile phone to 'roam' through the network. After the GSM system implemented roaming it was quickly duplicated by the other mobile phone systems.

Base Transceiver Stations

The GSM system works over a whole network of transceivers, spread across the world. Every transceiver is capable of accepting phone calls, text messages and, in recent years, internet connections. Transceivers are build up of 'cells', antenna's with a certain range and capacity. Every cell can handle about 40 phone calls at one time, and in busy cities transceivers are equipped with multiple cells, just to offer enough service for everyone. There are five different types of transceiver cells, based on their range:

- **Umbrella-cell** (Very long range)

  Umbrella-cells have the longest possible range found among transceivers, 35 kilometres. Umbrella-cells are often used for overlaying coverage, spanning the areas between two other cells where coverage is at its lowest. They are also used for highways and roads where people move at high speeds, when a lot of transitions between different transceivers are made. In that case the connection will be handed to the umbrella-cell, which will cause the number of transitions to go down dramatically.

- **Macro-cell** (Long range)

  Macro-cells are often located in the countryside, where there is only a small demand for coverage and therefore a few cells can give enough coverage to a large area. Each macro-cell has a range of 2 to 10 kilometres, based on the surroundings of the cell.

- **Micro-cell** (Medium range)

  Micro-cells are often found in towns and cities. They have a maximum range of about 2 kilometres, but rarely reach that. Micro-cells are often located on top of buildings, spread out over a city.

- **Pico-cell** (Short range)

  Pico-cells are small transceivers that are often put in places where the other transceivers cannot reach, for example in tunnels and large buildings. They are also used to increase coverage on a very busy point, like a train station. Pico-cells have a maximum range of 100 metres.

- **Femto-cell** (Tiny range)

  Femto-cells are the newest addition to the set of transceivers. They are designed for offices and houses, to increase the coverage inside buildings. That is why femto-cells only have a range of about 10 metres, and can only handle 10 to 16 phones at one time (for residential homes this is 2 to 4 phones).

Data from the GSM network

Mobile phones that are active (but not calling/messaging) will regularly send out a signal, telling the network where it is. It does this by connecting to the nearest transceiver and transmitting their SIM code. That transceiver updates the SIM code into a Visitor Location Register (VLR). That way, if another phone tries to reach that phone it only has to check the VLR to see if the phone is in that specific location.

Through different operators different data is stored in the VLR, which means that GSM datasets often differ from each other. Sometimes so much information is stored that the exact location of a mobile phone can be traced, other times only a broad area is known where the mobile phone could have been.

The data from mobile phones is already used in some projects, though not for OD matrix calibration. A prime example is the agreement between TomTom and Vodafone. Vodafone supplies TomTom with depersonalised information about their mobile phones, which TomTom uses in traffic modelling and congestion warnings.

2.4.2 Global Positioning System

GPS locators in mobile phones are a new ‘feature’, giving users the possibility to check their own location, and based on that location get recommendations for restaurants, public transport timetables and other useful information. But those locations are also stored in a database and can therefore be used for OD matrix calibration.

Background

The current GPS network is based on several older networks, for example LORAN, MOSAIC and Transit. GPS was originally devised by the US military to be able to track their own aeroplanes. Then in 1973 a group of twelve military officers devised a satellite system that would be used for keeping track of all defence personal. This satellite system was called Defence Navigation Satellite System, but was renamed to Navstar in the same year. Navstar was then renamed to Navstar-GPS, which was then shortened to just GPS.

As GPS was devised and constructed by the US military they were also the only ones able to use the system. This changed when in 1983 a Korean aeroplane flew into Russian airspace, due to navigation errors, and was consequently shot down. That event caused President Reagan to make GPS available for civilian use, when it was fully operational, in April 1995. Today the GPS system is still owned and operated by the US government, which reserves the right to deny the GPS service on a regional basis, in case of war and other conflicts.

GPS locators work by connecting with the satellites in orbit. Each satellite sends out a signal containing the time the message was transmitted, the precise orbital data of that satellite, and the status and orbit of all the other satellites. With data from several satellites the locator then calculates its own exact location.

Data from GPS locators

GPS locators take a certain amount of time to find their exact location. Because users might not want to wait, the locator keeps itself up-to-date, regularly checking its location. These updates are sometimes stored at the mobile phone provider, due to European and American law. These laws state that mobile communications providers have to be able to provide the exact location of their mobile phones in a certain amount of time. The reason for these laws is for the emergency numbers 112 and 911, when they are called the emergency service has to know where the caller is at that moment. That is why those restrictions were placed on mobile phone providers, and that is why they store GPS coordinates of their customers.

The GPS coordinates can be given in longitude and latitude, but that is not always the case. Sometimes the coordinates are given in another format, also called “datum”. These datums differ in the way the shape of the earth is modelled, and therefore how the round surface of the earth is converted to a 2-dimensional picture.

There are some projects that use GPS datasets for a variety of studies concerning traffic flows, decisions based on road-side information and modelling OD matrices. Most of these datasets however are generated by car navigation systems or specially placed GPS locators, and not with data from GPS locators in mobile phones.

2.4.3 Map Matching

With only one location of a mobile phone not much can be done. But when a whole series of these locations is available it becomes possible to find out which route that mobile phone took, and at which speeds the route was traversed. The route and speeds can in their turn give additional information about the origin, destination and also the mode of the trip that the mobile phone took. The process of finding a route from a series of locations is called map matching.

There has been a lot of research done in the area of map matching with respect to GPS data ([Quddus *et al.*, 2007]), but not really anything for GSM data. For some GSM datasets the same methods can probably be used, but this is not the case in general. The real difference between GPS and GSM data is that the GPS datasets contain actual coordinates, while GSM datasets have areas where the mobile phone could have been. This difference will mean that both types will have to be modelled and implemented in a different way.

In the next chapter an introduction will be given into OmniTRANS, the computer program that will be used for this thesis.

3. Introduction to OmniTRANS

In this chapter the software OmniTRANS will be introduced. In section 3.1 some background information about OmniTRANS and its history will be given. Section 3.2 introduces the dimension as they are used in OmniTRANS. Then in section 3.3 the structure of a network is given. Section 3.4 treats the trip distribution in OmniTRANS, with the implementation of the Gravity model. Section 3.5 will give two methods for the route assignment. Section 3.6 treats the calibration of OD matrices, and finally in section 3.7 the future gradient descend method will be given.

(Most information in this chapter comes from [Smits, 2010], [Omnitrans, 2010] and [OmniTRANS, 2010])

3.1 Background Information

According to their official site OmniTRANS is "Transport Planning Software" [Omnitrans International, 2010]. This means that it models "Transport" (displacements, people or goods moving from one location to another), then starts "Planning" (evaluating, assessing and (re)designing the network and data), combining both Transport and Planning in "Software" (a computer program).

OmniTRANS was originally designed and implemented by a company called Goudappel Coffeng, the largest Dutch Traffic and Transport consultancy. During the nineties Goudappel Coffeng noticed that the software that was in use for traffic and transport modelling was inadequate for their needs. Things that were unsatisfactory were for example a poor interface, lack of data management capabilities, limitations of the project sizes and the inability to (correctly) transfer model runs.

These limitations led to the decision to start the development of their own software, called OmniTRANS. The development of OmniTRANS took 9 months, and after it was offered to Goudappel Coffeng's customers, who were at the time using other software, almost all of them decided to switch to OmniTRANS. Due to the positive response, Goudappel decided to continue developing OmniTRANS. In May 2003 the popularity of OmniTRANS convinced Goudappel to market OmniTRANS internationally, and therefore formed the company Omnitrans International.

One distinctive feature of OmniTRANS is that it works very well in the area of data management (especially for large amounts of data). It also has a job-engine where users can define their own 'jobs', small program-like computations that can be performed on the model. A third aspect is the ability to also work with dynamic models, working with different time intervals and even with predictions of trends. OmniTRANS also has a clear user interface, and for expert users has a whole range of classes and features to improve modelling and solving.

3.2 Dimensions

In every computer program data that is entered has to be first divided into categories, for the structure of the data. These categories are also called dimensions, as data is sometimes entered as a hypercube (a multi-dimensional matrix), with the categories acting as dimensions. In OmniTRANS there are four standard dimensions; purpose, mode, time, and user-defined. For every trip that is generated in OmniTRANS these dimensions are given. Together the dimensions are often called PMTU, why will be explained after the dimensions themselves have been treated.

- **P***urpose*
  Purpose defines the objective of a trip. A purpose can be for example going ‘from home to work’, or going ‘from work to shopping’.
- **M***ode*
  Mode defines the type of transport. Two examples of modes are ‘car’ and ‘public transport’.
- **T***ime*
  Time is the time interval in which the trip takes place. A time interval can be for example just ‘from 10:00-10:15’, or the whole ‘AM peak’.
- **U***ser-defined*
  User-defined dimensions can be anything the user might want to implement. Examples of dimensions can be ‘car ownership’, or a differentiation of internal/external traffic.

Aside from these four dimensions another two dimensions appear in OmniTRANS, at the moment OmniTRANS has generated its own data. These two dimensions are:

- **R***esult*
  Result is the solution after OmniTRANS has finished a certain number of iterations in an algorithm, for instance a route assignment or an OD matrix calibration.
- **I***teration*
  Iteration is the current step in the calculations.

Together al six dimensions are called PMTURI. This is because the order of these dimensions is very important in OmniTRANS. If, for example, you are looking for the number of cars going to work at 6 pm, that can be written as [work,car,6pm,all]. OmniTRANS will then show a matrix with all cars, going from one centroid (vertical) to another centroid (horizontal). The same works for results, only then you will need six entries.

3.3 Modelling a Network

Before OmniTRANS can do any calculations a network has to be modelled. For the network first a map of the area is entered in OmniTRANS. This area is then divided into zones, and at the centre of each zone there is a centroid, a representative point of that zone. Each centroid has information about that zone, like the number or residents and jobs. Additional centroids are added for "the rest of the world", so that traffic that enters/leaves the area can also be modelled.

Centroids are connected to the road network, which is represented by nodes and links. Every road is represented by a link, a line in the network. Every link has certain properties like capacity, length and type (which mode of vehicle can use that link). In most models every link has a distinct direction, which means that two way streets have to be modelled by two links. But in OmniTRANS these are modelled by one link, with two sides that can be defined separately. Not all roads in a network will be represented by a link, small residential streets and dead-end streets will often be skipped.

At every point where two or more links meet a node is placed, connecting the links together. Nodes are mostly used as junctions (if there are more then two links attached), though not every junction in an area is modelled. As with real junctions there are several characteristics that influence the travel time, for example the approaching lanes (separate lanes for turning left/right), and the presence of traffic lights. These characteristics have an effect on the flow and speed of traffic and therefore have to be included in the model.

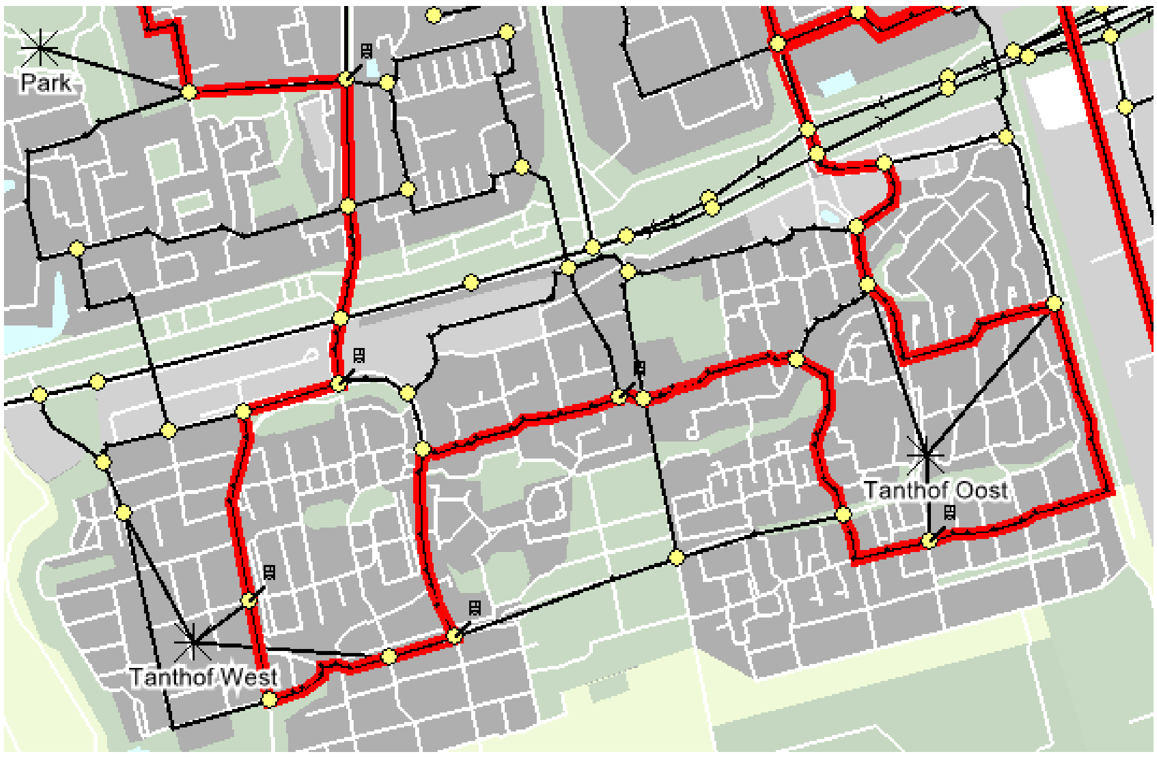


Figure 3.1: A Neighbourhood in Delft Modelled by OmniTRANS

3.4 Trip Distribution

Trip distribution is the process of linking origins and destinations together, to create trips. Trip Distribution is often done with a mathematical algorithm. In OmniTRANS the Gravity model is used, and therefore the implementation of that model into OmniTRANS will be discussed in this section.

In OmniTRANS there is a whole class of procedures for solving the gravity model: OtGravity. The OtGravity class first calculates the "production" and "attraction" of every centroid, based on the total number of trips leaving/arriving at any other centroid. But that means that only separate departures and arrivals are known. The goal is then to distribute these trips such that departures and arrivals are linked together.

OtGravity does the trip distribution with a doubly constrained model. On one hand it will balance the trips between each zone to the travel cost between the two zones. And on the other hand it will try to keep as close as possible to the original constraints created by the productions and attractions of each zone. OtGravity does this with an iterative approach, hopefully converging to a good solution. Because the productions and attractions are often not in balance (if there are in total more people leaving than arriving, or the other way around) there is also the option to either use only the attractions or only the productions.

The travel cost between two zones can be influenced greatly by the user. The travel cost can be dependent on time, distance, number of junctions, or any other generalized cost. Aside from the travel cost also the "perseverance" of travellers is needed. The perseverance will tell OtGravity how much cost the travellers are willing to 'pay' to get to their destination. With the measure of perseverance OtGravity will be able to distribute trips over the different zones, with a distribution function. There are at the moment four functions implemented in OtGravity:

Log-normal	$F_v(z_{ijv}) = \alpha_v \cdot e^{\left(\beta_v \cdot \ln^2(z_{ijv}+1)\right)}$
Top Log-normal	$F_v(z_{ijv}) = \alpha_v \cdot e^{(\beta_v \cdot \ln(z_{ijv}/\gamma_v))}$
Exponential	$F_v(z_{ijv}) = \alpha_v \cdot e^{(\beta_v \cdot z_{ijv})}$
Defined by a discrete array	$[[x_1, x_2, .., x_n], [y_1, y_2, .., y_n]]$

In the above formulas F_v is the distribution function for mode v, z_{ijv} is the impedance (travel cost) between centroids i and j for mode v, and α, β and γ are parameters.

3.5 Route Assignment

As mentioned in Chapter 2 two methods for the assignment of traffic flows from the OD matrix will now be discussed. A reminder: the "assignment of **g**" is distributing the trips over the network, finding a path for every trip. The assignment of **g** not only influences the travel time and distance, but also the appearance of congestion. The two methods that will be treated, as they are implemented in OmniTRANS, are All or Nothing and Volume Averaging.

3.5.1 All or Nothing (AON)

All or Nothing is the simplest route assignment possible. First the shortest path between every zone is calculated, there are several different algorithms that do that. In OmniTRANS a reverse propagation algorithm is used, starting at a destination and working backwards until it has reached every other zone. After the shortest paths have been found the assignment is simple: just sent all traffic across that path. Therefore an AON assignment is actually the first step in a 'normal' user equilibrium computation.

Of course there are two very big downsides to the AON approach. The first downside is that it can happen that the assignment through the network can exceed the capacity of one or more links in the network. This is of course impossible in practice. The second downside is that an AON assignment doesn't take congestion into account. This means that, even if the capacity is not exceeded, the travel time can go up, just because there are too many vehicles on the road. And therefore trips will actually use a different route, avoiding the congestion.

Despite these two downsides the AON assignment can give good results in small networks where congestion isn't a problem. It is also very useful to give a first impression of a network, to see if it is going to be congested or not (exceeding capacities is a good indication of congestion).

3.5.2 Volume Averaging (VA)

Volume averaging is a little bit more complex compared to AON, but it gives much better results. VA works with volumes, which is just the amount of traffic flow on a link. The assignment itself is done with an iterative process: the volume in one iteration is dependent on the volume from the last iteration. In the first iteration the volumes found by an AON assignment are used, for the other iterations the following formula is used:

$$V_l^n = (1-\phi)V_l^{n-1} + \phi F_l \qquad\qquad (0 \leqslant \phi \leqslant 1)$$

with V_l^n the volume on link l in iteration n, F_l the volume on link l from the AON assignment, and ϕ indicating how close the solution should stay to the AON assignment. ϕ can be a constant or change depending on the number of iterations.

3.5.3 Travel Time Functions

In the VA assignment also the travel time is important, because it is influenced by the volume of traffic on a link (congestion causes traffic to slow down). That is why in the VA assignment also the relationship between volume and travel time is defined. In OmniTRANS three different functions are supported; the BPR-function (from the Bureau of Public Roads), the gradual linear function, and the continuous function. From these three the BPR-function is the most commonly used.

The BPR-function works with the following formula:

$$T_l = T_l^0 \left(1 + \alpha \left(V_l/Q_l\right)^{\beta}\right)$$

where T_l is the travel time on link l, T_l^0 is the travel time without the effects of congestion, V_l and Q_l are the volume and capacity of link l, respectively, and α and β are parameters of the model.

3.6 OD Matrix Calibration

For the calibration of OD matrices OmniTRANS has created a specific class: OtMatrix-Estimation. The OtMatrixEstimation class contains several techniques that can help with calibrating the OD matrix. In the process of calibrating in OmniTRANS also other information is used and changed, to improve the whole model. Other things that might be calibrated are the mode distribution, route choices, departure times and many other small variables.

OtMatrixEstimation uses information from a variety of sources such as traffic counts, home interviews, on-board (public-transit) surveys, etc. How these sources are used will be explained in the following section.

3.6.1 Data Sources

(Information in this section comes from [Schilpzand, 2009])

Matrix calibration works by looking for OD flows, cells from the matrix **g**, which 'differ' from reality, and multiplying these with a compensation factor. The amount of 'difference' is determined by the restrictions on the OD matrix. These restrictions are the implementation of the data sources that define the observation of the real OD matrix. In OmniTRANS the data sources are implemented in four different ways: counts, screenlines, blocks and trip ends.

- **Counts**
  Counts are the most basic input type of traffic counts. A 'count' in OmniTRANS is a representation of the most simple traffic count: counting all the cars that pass over a road. A count therefore only tells something about that particularly link, and nothing about the rest of the network.

- **Screenlines**
  Screenlines are a bit more complicated than counts. Each screenline is a combination of counts, grouping the counts together for the calibration. Screenlines are useful in case of parallel roads, for instance a highway and a parallel section for local traffic. Then the counts on those roads can be taken together. A screenline itself has no value, the total value is just the sum of all selected counts.

- **Blocks**
  Blocks work differently than counts. If a part of the OD matrix is known, for example from a travel survey, it can be implemented in a block. A block means that the known cells are calibrated together, with the total value of the block. A small example:

	1	2	3	4	5	6	7	⋯
1	-	6	8	0	9	1	1	
2	3	-	1	5	7	6	2	
3	5	2	-	1	0	7	4	
4	0	1	8	-	5	5	3	
5	1	2	7	5	-	6	4	
⋮								

Figure 3.2: Example of a Block

  In the above example all the amount of traffic between centroids 1, 2, 3, and centroids 4, 5, 6 is known, these entries are coloured in grey. The block then also holds the total amount of traffic, and this is taken into account in the calibrations. For the example the block might have a real value of 40, while the cells in the OD matrix only have a value of 36.

- **Trip Ends**
  Trip ends are exactly what the name implies: the ends of a trip (the end can be either the beginning/origin or the end/destination of a trip). If the amount of people arriving at a zone is known, that information can be used to calibrate a whole column of the OD matrix, $\sum_i g_{ij}$. The same holds for departing people and a row of the OD matrix, $\sum_j g_{ij}$:

	1	2	3	4	5	6	7	⋯
1	-	6	8	0	9	1	1	
2	3	-	1	5	7	6	2	
3	5	2	-	1	0	7	4	
4	0	1	8	-	5	5	3	
5	1	2	7	5	-	6	4	
⋮								

Figure 3.3: Example of a Trip End

3.6.2 Formulas

Counts (and therefore also screenlines) cannot be used for the calibration of OD matrices as they are. A count is only defined on a link in the network, but links do not appear in an OD matrix. To connect separate links to an OD pair from the OD matrix, so-called screenline matrices are created. These shouldn't be confused with screenlines, despite the similarity in name.

Screenline matrices are made for each link l, and hold information for every OD pair. Each value of a cell (i, j) lies on the interval $[0, 1]$, and indicate which percentage of the total flow going from centroid i to j actually goes across link l. For example:

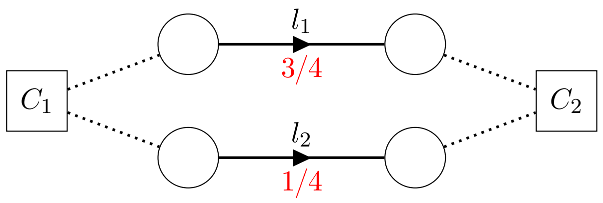


Figure 3.4: Example Network

The above example network gives rise to the following screenline matrices:

$$S_{l_1} = \begin{bmatrix} - & \frac{3}{4} \\ 0 & - \end{bmatrix} \text{ and } S_{l_2} = \begin{bmatrix} - & \frac{1}{4} \\ 0 & - \end{bmatrix}$$

Then OD matrix calibration works by checking every restriction, one after another, and 'implementing' them to the starting matrix. A 'restriction' is the general term for the data sources mentioned above; counts, screenlines, blocks and trip ends. For each restriction all OD pairs are calibrated, by multiplying their original value with a calibration factor χ_0:

$$\chi_0 = \frac{\sum_r C_r}{\sum_r (g_{ij})_r}$$

Here C_r is the value of the rth restriction, and $(g_{ij})_r$ the cells in the matrix belonging to the rth restriction.

For the actual calibration every restriction is calibrated separately. Each restriction calculates a calibration factor, based on the following formula:

$$g_{ij}^{new} = g_{ij}^{old} \cdot \left(\frac{C_r}{\sum\limits_{i,j \in r} P_{ijr} g_{ij}^{old}} \right)^{\epsilon_r}$$

with g_{ij}^{new} and g_{ij}^{old} the new and old values of cell (i, j), C_r the total value of restriction r, P_{ijr} the fraction of the OD pair (i, j) passing restriction r, and ϵ_r the elasticity of restriction r (how 'reliable' the restriction is).

In the above formula there is no direct mention of $\mathbf{v}$ or $\hat{\mathbf{v}}$. The value of $\hat{\mathbf{v}}$ is of course the value C_r, and the original value of $\mathbf{v}$ is calculated by the sum $\sum_{i,j \in r} P_{ijr} g_{ij}^{old}$. For this calculations only the OD flows that are used by the restriction are used: $\{i, j \mid \mathbf{v}(g_{i,j}) \in r\}$.

In OmniTRANS these four data sources are given different priorities, which can be set by the user. By default the order of these sources is: Blocks → Counts → Screenlines → Trip Ends.

3.7 Gradient Descend Method

As mentioned in Chapter 2 this thesis also looks at the gradient descend method to solve the problem of calibrating an OD matrix. In particularly the gradient descend method as given by [Smits, 2010] will be used. This version was designed especially for OmniTRANS, and although it is not yet implemented in OmniTRANS, the planning is that this will be done in the near future.

From the literature we have the following general problem description for OD matrix calibration:

$$\begin{aligned}
\min_{\mathbf{g},\mathbf{l}} F(\mathbf{g},\mathbf{l}) &= \alpha F_1(\mathbf{g},\hat{\mathbf{g}}) + (1-\alpha)F_2(\mathbf{l},C) \\
s.t.\ \ \mathbf{l} &= assign(\mathbf{g}), \\
\mathbf{g} &\geqslant 0, \\
\mathbf{l} &\geqslant 0.
\end{aligned} \tag{GP}$$

In the above description the formulation of [Smits, 2010] is already used. Here $\mathbf{l}$ is the set of all flows (which used to be $\mathbf{v}$), and C the set of all counted values (which used to be $\hat{\mathbf{v}}$).

[Smits, 2010] uses for the cost functions F_1 and F_2 the L_2 norm, after considering several alternatives. The factor ϵ_r is reused, to give a weight to each restriction in the objective function. The above problem is still a bi-level optimization problem (both the assignment of $\mathbf{g}$ and the general problem is an optimization problem). Which means that it is difficult to solve numerically, and often no good solution is found at all.

To combat this drawback [Smits, 2010] assumes that the assignment of $\mathbf{g}$ is locally proportional, meaning that between iterations the route assignment does not change much. Therefore, with that assumption, the assignment of $\mathbf{g}$ doesn't have to be calculated every iteration, which makes the problem a single level convex sub-problem of the original:

$$\begin{aligned}
\min_{\mathbf{g}} F(\mathbf{g}) = \alpha \sum_{\substack{(i,j)\in I \\ d\in D}} \frac{1}{2}\left(g_{ij}^d - \hat{g}_{ij}^d\right)^2 + (1-\alpha)\sum_{r\in R}\frac{\epsilon_r}{2}\left(\sum_{\substack{(i,j)\in I, \\ d\in D_r}} P_{ijr}^d g_{ij}^d - C_r\right)^2 \\
\text{s.t.}\ \ \mathbf{g} \geqslant 0
\end{aligned}$$

In the above formulation D is the set of dimensions (as defined by OmniTRANS, section 3.2), D_r is the set of dimensions used by restriction r, I the set of all OD pairs i, j, and P_{ijr}^d is the fraction of the demand of OD pair (i,j) in dimension d that applies to restriction r. The variables $\mathbf{v}$ and $\hat{\mathbf{v}}$ (or $\mathbf{l}$ and C) are considered implicitly; C_r is the value of $\hat{\mathbf{v}}$, and $\sum P_{ijr}^d g_{ij}^d$ is the value of $\mathbf{v}$.

From the gradient descend method [Smits, 2010] devised the following algorithm:

Algorithm The OmniTRANS gradient method algorithm

```
1 : $stopCrit \leftarrow \textbf{false}$
2 : $\mathbf{g} \leftarrow \tilde{\mathbf{g}}$
3 : while not $stopCrit$ do
4 :     $\mathbf{s} \leftarrow -\mathbf{g} \cdot \nabla F(\mathbf{g})$ [1]
5 :     $\lambda^* \Leftarrow \frac{\partial}{\partial \lambda} F(\mathbf{g} + \lambda s) = 0$ [2]
6 :     $\mathbf{g} \leftarrow \mathbf{g} + \lambda^* \mathbf{s}$
7 :     update $stopCrit$
8 : end while
9 : $\mathbf{g} \leftarrow \mathbf{g}^+$ [3]
```

Steps [1] and [2] can be solved algebraically, giving a formula for the solution of $\mathbf{s}$ and λ^*, respectively. These calculations or the results will not be shown here, for more detail please consult [Smits, 2010]. In step [1] also the gradient $-\nabla F(\mathbf{g})$ is multiplied with the current OD matrix $\mathbf{g}$. This is done to make sure that OD pairs that are 0 remain so. Some comments on this approach will be discussed in section 15.2.

One remark must be made concerning [3], given its strange appearance. Because the algorithm itself does not guarantee a positive value for all g_{ij}, it has to be forced with an additional step:

$$\mathbf{g}^+ = \left\{ \max \left\{ 0, g^d_{i,j} \right\} \mid (i,j) \in I, d \in D \right\}$$

The above is of course an ugly solution. According to [Smits, 2010] $\mathbf{g}$ never became negative in the test cases that were used, so it is assumed that this situation will seldom occur. In section 15.2 some more comments on this problem will be given.

4. Problem Specification

With the Literature Research done, a lot of different information passed in review. Therefore it is now time to give some more specification of the problem that will be treated in this thesis. In section 4.1 something will be said about the OD matrix calibration in general. Then section 4.2 treats the restrictions that will have to be formulated. And in section 4.3 a look will be taken at routes generated by the data.

4.1 OD Matrix Calibration

As mentioned in section 2.2.2 there are several different solution methods for the problem of OD matrix calibration. At the moment OmniTRANS uses a custom made algorithm, which works by multiplying OD pairs with a factor based on the restrictions added to OmniTRANS (counts, screenlines, blocks and trip ends, as defined in section 3.6). This calibration does not work as good as the people at Omnitrans want, and therefore in the first part of 2010 a large study was made of OD matrix calibrations.

From that study two new algorithms were found, which are described in depth in [Smits, 2010]. The first algorithm is an improved version of the custom algorithm in OmniTRANS, the second is a new algorithm based on a gradient descend method. These two new algorithms are not yet implemented in OmniTRANS, but the implementation is planned for the near future. In studying the two algorithms two small errors were found, and these will be changed before the algorithms will be used for this thesis.

The adapted OmniTRANS algorithm uses the same calibration factors as described in section 3.6, but for every OD pair all calibration factors are taken together, by taking the geometric mean of the factors. Because not all factors should be weighted equally, the geometric mean is weighted with P_{ijr}, the percentage of the OD flow passing the restriction. According to [Smits, 2010] this gives the following formula:

$$g_{ij}^{new} = g_{ij}^{old} \cdot \left[\prod_{r \in R} \left(\frac{C_r}{\sum\limits_{i,j \in r} g_{ij}^{old} \cdot P_{ijr}} \right)^{\epsilon_r P_{ijr}} \right]^{\frac{1}{\sum\limits_{r \in R} \epsilon_r P_{ijr}}}$$

But the above formula is not entirely correct. The calibration factors are weighted with the percentage P_{ijr}, but in the exponent the sum is taken of $\epsilon_r P_{ijr}$, which is incorrect. This is easily seen when there is only one calibration factor. In that case the elasticity in the calibration factor is cancelled by the elasticity in the exponent.

Therefore the real formula should be:

$$g_{ij}^{new} = g_{ij}^{old} \cdot \left[\prod_{r \in R} \left(\frac{C_r}{\sum\limits_{i,j \in r} g_{ij}^{old} \cdot P_{ijr}} \right)^{\epsilon_r \cdot P_{ijr}} \right]^{\frac{1}{\sum\limits_{r \in R} P_{ijr}}}$$

For the gradient descent method another (small) improvement was found. In [Smits, 2010] the concern was that with the gradient descend method it could happen that cells in the OD matrix became negative. His solution was to truncate all negative cells to 0, at the end of the calibration. But that means that perhaps not the optimal **positive** solution was found. Therefore it is better to check the non-negativity in the algorithm itself, leading to the following algorithm:.

Algorithm The OmniTRANS gradient method algorithm (2)

```
1 : stopCrit ← false
2 : $\mathbf{g} \leftarrow \tilde{\mathbf{g}}$
3 : while not stopCrit do
4 :     $\mathbf{s} \leftarrow -\mathbf{g} \cdot \nabla F(\mathbf{g})$
5 :     $\lambda^* \Leftarrow \frac{\partial}{\partial \lambda} F(\mathbf{g} + \lambda s) = 0$
6 :     $(\lambda^* \mathbf{s})_{ij} \leftarrow \max\{-g_{ij}, (\lambda^* \mathbf{s})_{ij}\} \ \forall_{i,j}$
7 :     $\mathbf{g} \leftarrow \mathbf{g} + \lambda^* \mathbf{s}$
8 :     update stopCrit
9 : end while
```

More discussion on this aspect of the gradient method will be treated in Chapter 15.

With the solution method(s) of the OD matrix calibration given, it means that in the rest of this thesis only attention will be given to the data from the public transport and from mobile phones. This data will have to be modelled and converted to a format which can be used for OD matrix calibration.

4.2 Restrictions

The way that information about the real world (traffic counts, surveys, etc.) is used by OD matrix calibration is with so-called restrictions. Restrictions have a value (C_r) and weight (ϵ_r), and a list of OD pairs (i, j) passing that restriction. And because not always the entire flow of an OD pair passes the restriction, also for every OD pair the fraction of flow that does pass the restriction is known (P_{ijr}). This also makes it easy to define a general definition: if an OD pair is not used by the restriction then $P_{ijr} = 0$.

For the two algorithms treated above (Improved OmniTRANS and Gradient Descent) the restrictions are taken into account in two different ways. For the adapted OmniTRANS algorithm the restrictions are taken into account with the calibration factors, $\frac{C_r}{\sum_{i,j} g_{ij}^{old} \cdot P_{ijr}}$. For the gradient descent method the restrictions are part of the objective function F_2, by the distance between $\mathbf{v}$ and $\hat{\mathbf{v}}$. Therefore the restrictions are taken into account when calculating the optimal direction s, and the step length λ^*.

At the moment there are four types of restrictions formulated in OmniTRANS: counts, screenlines, blocks and trip ends. These four types are enough when working with information from traffic counts and surveys. But the expectation is that these will not be enough for this thesis. Because the data from the OV-Chipkaart and from mobile phones give more information than traffic counts, more flexible restrictions will be needed. But this is something that will be further discussed in the modelling chapters of the OV-Chipkaart data (Chapter 5) and the mobile phone data (Chapter 10).

4.3 Routes

Something that has not been mentioned before, but is of some importance, is that the data of mobile phones can perhaps generate more data than just an OD matrix calibration. A dataset from a mobile phone provider shows the actual route that the mobile phone took, not just an origin and destination. Therefore it would be a waste not to use that information. Using the routes from mobile phone datasets is of course no longer part of OD matrix calibration, and therefore it will be studied how routes can be best inserted into the research of this thesis.

With the above comments in mind the real modelling will now begin. In Part I the modelling of the Public Transport will be given, and in Part II the modelling of mobile phones.

Part I

Public Transport

5. Modelling OV-Chipkaart data

Now that the background of the project has been treated it is time for the modelling phase. In this chapter a start will be made with modelling the OV-Chipkaart data. In section 5.1 first something will be said about how public transport data is used in the literature. Then in section 5.2 a look will be taken into the structure of the data from the OV-Chipkaart. And in section 5.3 the first step will be taken in the modelling, resulting in three different approaches.

5.1 Other Literature

In the Literature Review (Chapter 2) several publications were found dealing with the problem of OD matrix calibration. Most of these publications however made no reference to using data from public transport. In the calibration of OD matrices mainly traffic counts are used, and traffic counts are only available for cars, not for public transport passengers. Of course some publications have such a general problem description, that public transport could easily be included. This general description however doesn't give any insight as to how public transport data should be structured or used. Therefore not much is known about using public transport data for the calibration of OD matrices.

Two papers that do explicitly use public transport data for OD matrix calibration are: [Cui, 2006] and [Chan, 2007]. In these papers the OD matrices were calibrated with information from the London Underground and the Chicago bus network, respectively. The big difference between those papers and the situation used in this thesis is that they use *only* data from the public transport, they model the transit lines as a closed circuit without any influence from other modes. This has as effect that the stops (stations) are modelled as the zones, and therefore all data from one stop to another is taken into account, as it is exactly the traffic between one OD pair.

The above situation is of course not the case in the model as it is used in this thesis. In this model the public transport is part of the normal road network, and therefore is influenced by congestion. Also, as the zones are based on a whole neighbourhood and not a single stop, not all information can be used directly. Some information has to be taken together, when one path between an OD pair crosses several transit lines, and some information has to be spread out, when multiple OD pairs have a path using a specific transit line, for example a bus line. Therefore these papers cannot be used.

Without any pre-existing research to build on, the whole modelling phase will begin with an examination of the data structure, as it is provided by the OV-Chipkaart.

5.2 Data Structure

The data generated by the OV-Chipkaart is the collection of all journeys made by people in the public transport. Every time a person checks in at the start of his or her journey the starting stop is stored, and when the person checks out again the final stop is added, creating a whole journey. The collection of all these journeys is of course aggregated, for a certain time period, to remove any personal information.

There are three types of journeys that are stored by the OV-Chipkaart:

- A complete journey
- Zero travel distance
- Only a check-in

From these three only the complete journey is of use. The zero travel distance implies that a person entered a station and left again, without actually travelling through the network. Zero travel distances happens for example when people come to collect someone. As no actual journey is made, these results are ignored.

With only a check-in the destination of a journey is unknown. Only a check-in happens if a person forgets to check-out at the final stop, or checks out without having checked in (in that case the system considers it a check-in). If the destination of a trip is missing not much can be said about that trip. There have been publishings trying to find the whole trip, but the results vary per situation. And as the assumption is that this will not happen very often, these types are also ignored.

The results from the complete journeys are aggregated, and give rise to a large matrix H, showing the number of journeys from one stop to another in a certain time period (for example one hour). To illustrate: the cell $H(f, k)$ contains the number of people checking in at stop f, and checking out at stop k. The cells $H(f, f)$ are all empty, as zero distance journeys are ignored. Because the results in the matrix H are aggregated already some information is lost. For example a transfer to another transit line counts as two separate journeys, and therefore the information about the entire journey is lost, and only the separate parts are known.

The big benefit of using data from the OV-Chipkaart is of course that it is collected every second of every day. Therefore the numbers are not dependant on one survey, but on numbers from a longer time interval. Which means that fluctuations are reduced in the data, and the trend of the data is shown, instead of a snapshot of one particular instant.

There is one important aspect that influences the quality of OV-Chipkaart data, namely the fact if the OV-Chipkaart is the only valid payment option, or if there are still other methods. If the OV-Chipkaart is the only valid payment option the data is complete, it shows all journeys made with the public transport. But in the case that there are still other valid payment options, the data becomes incomplete, as not all journeys are tracked. In that case another approach must be used, to be able to use the data.

5.3 Initial Modelling

With the structure of the OV-Chipkaart data defined, a begin can be made with the modelling of OV-Chipkaart data.

5.3.1 Converting to Counts

In OmniTRANS all traffic data that is used for OD calibration is entered as a count on a link. Therefore the first step in the OV-Chipkaart modelling is to see if the definition of counts can also be used for OV-Chipkaart data. The information in matrix H is of course based on stops, not on links. When all transit lines and their stops are known (and it is assumed that this is the case), the numbers from the matrix H can be added to create a count for every link between two stops. For the calculation of counts an easy algorithm can be written (see section 6.2) with which this could be done.

If all data from the matrix H is converted to counts, it means that a lot of information is lost. With counts it is only known that a certain number of people passed that link, but the data from the matrix H also says something about where passengers came from and where they went after. It also means that that particular count is used to calibrate all OD pairs that have a path crossing it. While perhaps if the full journey was known, only one OD pair would fit that journey, and therefore the other OD pairs were wrongly calibrated.

These objections mean that OV-Chipkaart data cannot *just* be converted to counts, something else has to be done to preserve the extra information from the matrix H.

5.3.2 Routes

The second approach to modelling the OV-Chipkaart data is to model all the journeys as a count on multiple links. That way only journeys between OD pairs that use all links in a specific journey will be calibrated with the count belonging to that journey. These counts on several links will be called Routes, and can be easily modelled to be exactly a journey between two stops. Of course only Routes for which matrix H has any data will be taken into account.

Routes can be used in the same way as counts, as they are defined in section 3.6. The only difficulty lies in finding the OD pairs that use the *whole* Route. Finding these OD pairs consist of checking the route assignment of **g**, comparing the different journeys from the assignment with the journey in question.

Though the usage of Routes decreases the amount of information lost, it doesn't entirely remove it. In large networks there might be several stops on the journey between an OD pair, meaning that there will be several Routes that will be used to calibrate that OD pair. But not all those Routes belong to that OD pair, as can be easily seen in the following example:

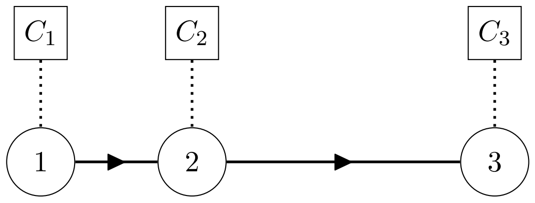


Figure 5.1: Two Routes that overlap

Between these stops there are two Routes: Route 1 goes from stop 1 to stop 3, with a count of 1, and Route 2 goes from stop 2 to stop 3 with a count of 20. From the above figure it is immediately apparent that Route 1 belongs to OD pair C_1C_3, and Route 2 belongs to OD pair C_2C_3. But in the calibration both Routes will be used for OD pair C_1C_3, resulting in a faulty result.

The above problem comes from the fact that the Routes are not modelled on public transport journeys. Routes do not take into account that there has to be a boarding at the first stop, and an alighting at the second stop. Therefore in the next section an approach will be given more suited to public transport.

5.3.3 Tracks

Before treating the new approach, first a look will be taken at a public transport journey. A public transport journey does not consist of just starting at one point, going along a route and ending at another point. Journeys made by public transport are build up of several 'sub-journeys'. First an access-journey is made, from the origin to the first stop. Then a public transport vehicle is boarded, and the journey continues along the transit line. During the journey it might happen that a transfer is made, splitting the journey. A transfer can be made either at one stop (alighting, and immediately boarding another transit line), or between two stops (alighting, walking to another stop, and boarding a new transit line). Finally, at the last stop there is still an egress-journey, going from the stop to the final destination.

In the situation sketched above it is of course strange to look at all journeys between OD pairs using the Route. For example if the passengers do not board at the first stop of the route, but one stop earlier, or alight at a stop much farther away, the Route is based on a totally different journey. Therefore a Track is defined as the transit journey starting at the first stop, and ending at the second stop, without any transfers.

With the above definition a Track is almost complete. The reason it is not entirely complete comes from the fact that with train journeys the journey can have a transfer, which does not lead to a check-out and check-in. This comes from the fact that the train system is actually one giant transit line, but it cannot be modelled as such. Therefore the restriction of no transfers is relaxed if the matrix H is defined for trains, and instead it is said that the used transit line(s) have to be of the correct transit provider (for example: NS). Therefore it is assumed that it is known for which transit provider the matrix H is defined.

A Track of course doesn't have to start immediately after an access-journey, or end at an egress-journey. A Track can also start at a transfer, or end at another transfer. This makes no difference for the modelling, as the Track will still start with a boarding and end with an alighting.

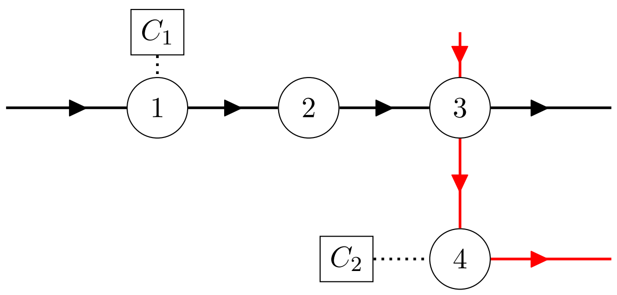


Figure 5.2: Example of two Tracks

In the above figure there are two Tracks: one going from stop 1 to stop 3, with the black transit line provider, and a second one going from stop 3 to stop 4, with the red transit line provider.

To find the OD pairs that actually use a Track more information will be needed from the route assignment than with counts. Earlier it was enough to know which paths crossed the specific link(s). But with Tracks also information is needed on the number of people boarding and alighting per OD pair, and with which transit line the specific boarding and alighting were made. This means that the route assignment will have to be more detailed than was needed before.

5.3.4 Connecting Tracks

After defining Tracks, the next logical step is of course to take a look at connecting different Tracks. For example, if there are 5 people going from stop 1 to stop 2, and 10 people from stop 2 to stop 3, how many of those people actually went from stop 1 to stop 3, transferring at stop 2? If that number is known these people can be modelled directly, instead of with two separate Tracks, which would improve the quality of the calibration.

The matrix H sadly does not hold any information on transfers. The data in matrix H is aggregated, removing any personal information, which means that the connection between different journeys is lost. Therefore the information about transfers have to come from another source, which is quite difficult to come by.

In the Netherlands the OV-Chipkaart data is gathered by Trans Link Systems, a company established by the five largest public transport companies in the Netherlands. Trans Link Systems is also the main supervisor of the OV-Chipkaart and the implementation of it. Trans Link Systems therefore receives all transactions generated by the OV-Chipkaart. It uses OV-Chipkaart data to check for illegal and hacked cards, so they can be disabled. Aside from Trans Links Systems, all public transport providers receive data on the transaction of the OV-Chipkaart in their own services. But that data is again aggregated, like the matrix H.

Now, if a transfer happens between two different public transport lines only Trans Link Systems has information on that transfer. But as they do not have any interest in modelling the transit flows of the OV-Chipkaart, and will never release any of their data, these kind of transfers will never be modelled.

5.3.5 Stops

It is also possible to examine the network on a stop-level, taking one stop and checking incoming and outgoing traffic. This traffic is visualised in the following figure:

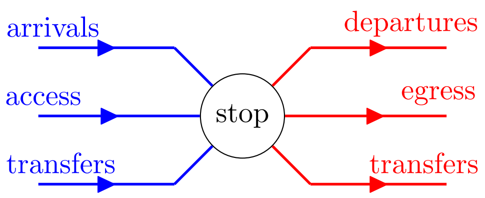


Figure 5.3: Flows in/out a stop

In the above figure the following flows are shown:

- **Arrivals/Departures**
  Arrivals and Departures are the passengers travelling to or from the stop, by means of a transit line. Passengers really start/end their journey here, passengers passing through without transferring are not modelled.

- **Access/Egress**
  Passengers walking from their origin to the stop to start their journey, or passengers walking from the stop to their destination.

- **Transfers**
  Transfers can happen in two ways. Transfers at the stop itself are included in the arrivals/departures. Transfers where the passenger has to walk from one stop to another is modelled separately, in this flow.

The transit flows in and out a stop are of course given by the matrix H. The transit flow going to a stop k is the sum of the kth column of H: $\sum_f H(f,k)$. The transit flow leaving a stop f is the sum of the fth row of H: $\sum_k H(f,k)$.

As can be seen by the above formulas using stops again aggregates the information from matrix H. This means that some information is lost, as now either the origin or the destination of a flow is not taken into account. But hopefully it will also mean that the calculations will be very fast, and in situations where the OD matrix already is close to the real world this additional information is not needed. The modelling of stops is therefore assumed to be a kind of 'fine-tuning'.

In the next chapter now several algorithms will be introduced, which will be needed for the modelling of the OV-Chipkaart data.

6. OV-Chipkaart Algorithms

Converting the data from the OV-Chipkaart cannot be done by hand, as that is far too much work. Therefore in this chapter several algorithms will be introduced, to convert OV-Chipkaart data to a format that can easily be used for OD matrix calibration. In section 6.1 first the variables that are needed for the algorithms will be introduced. Then in section 6.2 an algorithm will be given for creating counts, in section 6.3 an algorithm will be given for Tracks, and finally in section 6.4 an algorithm will be given for the stop modelling.

6.1 Variables

The following variables will be needed for the definition of the algorithms. The main variables are variables that belong to the network, and are therefore also used in the literature. The algorithm specific variables are variables especially defined for the algorithms in this thesis.

Main variables:

centroids:	$i,\, j \in C$,	with $\|C\| = c$		links:	$l \in A$,	with $\|A\| = a$
stops:	$f,\, k \in S$,	with $\|S\| = s$		transit lines:	$o \in T$,	with $\|T\| = t$

Algorithm specific variables:

H	is the matrix containing all OV-Chipkaart data; $H(f,k)$ is the number of people going from stop f to stop k
N_f^{before}	is the set of stops coming before stop f, on any transit line
N_f^{after}	is the set of stops coming after stop f, on any transit line
$paths(i,j)$	is the collection of all transit paths between centroids i and j
P_{ijp}	is the percentage of flow going over path $p \in paths(i,j)$
$TL^p(l)$	is the transit line attached to link l, on path p

6.2 Creating Counts

The following algorithm transforms the data from matrix H into counts (as used in Omni-TRANS) on the link between stops A and B. The algorithm works by checking all stops before A, and adding their flow to stops coming after B. If there are multiple transit lines passing both stops, it is checked that only flows connected by one transit line are taken into account.

In the following algorithm a count is created between stops A and B. Therefore the stops A and B are the input of the algorithm, and the output of the algorithm is a counter, signifying the value of the count that is to be created:

Algorithm 1: Creating Counts

```
1 : counter = 0
2 : for all $f \in N_A^{\text{before}}$ do
3 :     for all $k \in N_B^{\text{after}}$ do
4 :         if $f, k$ are connected, passing $AB$
5 :             counter = counter + $H(f, k)$
6 :         end
7 :     end
8 : end
```

In Algorithm 1 there are two for-loops, going through a list of stops, and one if statement. The maximum size of the stop lists is $(s-1)$, and checking if two stops are connected takes t steps, giving a complexity of $O(s^2 \cdot t)$. This complexity is of course a very large upper bound, as it will only be reached if the network is composed of overlapping circular transit lines. In practice the bound will be in the order of $O(e^2 \cdot t)$, with e the maximal number of predecessors or successors of a stop on any transit line.

Algorithm 1 is not yet perfect, as some counts could be counted twice. This can easily be seen in the following example network:

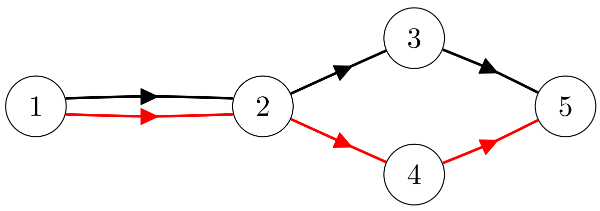


Figure 6.1: Counts taken twice

In the above figure the link (2,3) will count all passengers from 1 to 5, as they are both part of the black transit line. But some of these passengers will travel through stop 4, with the red transit line. Because the results in matrix H are already aggregated there is no clear solution for this problem. One option is to connect the counts (2,3) and (2,4), and the counts (3,5) and (4,5) with a screenline, such that they are taken together. But in any case, it will not have any influence on the complexity of the algorithm itself.

6.3 Finding OD pairs with Tracks

As mentioned in section 5.3.3 a lot more information from the route assignment is needed for checking which OD pair uses a Track. Now first a look will be taken into how much information is actually needed.

For the definition of a Track the following is needed: for every OD pair it should be known if (part of) the flow boards at stop f, and alights at stop k, both from the same transit line (or from the same transit line provider, in the case of trains). A first approach would therefore be just to check stops f and k, to see if there were any boardings or alightings. But this approach can quite easily be wrong, as is shown in the following example:

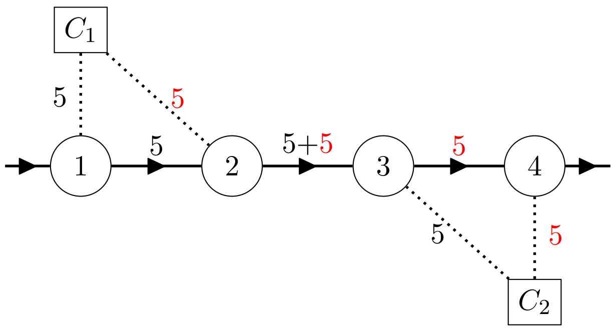


Figure 6.2: Two OD flows

The above approach would see that 5 people boarded at stop 1, and also 5 people alighted at stop 4, with the same transit line. But none of these 5 persons overlap, and therefore the OD pair (C_1,C_2) shouldn't be part of the Track generated by stops (1,4). In larger networks, with a lot of centroids and different transit lines and providers, the problem will only become more complex. Therefore more information is needed still.

A next step would be to look at the paths. If all paths between the centroids are known, for example with a list of all links that were traversed, a lot more can be said. With the path known it becomes possible to link the two stops. If they both appear in the path, it is known that the OD pair passed by the stops.

But it is still not known if the passengers actually boarded/alighted there. Therefore for every link the transit line should be given (with a 0 if no transit line was used). Then it becomes a matter of checking the transit lines of the two links attached to a stop. If the transit lines differ, there was a boarding, alighting or transfer, and the OD pair is valid. Of course, the links between the two stops also have to be checked, to see if there were no transfers during the journey.

The above approach is formulated in Algorithm 2. For the input a lot of data is needed. First off the set of centroids C and the set of paths between any two centroids, $paths(i, j)$, have to be known. Also for every path p the percentage of flow should be known, P_{ijp}. For every link l in a path p the transit line of that link is $TL^p(l)$. The algorithm then calculates the percentage of flow for every OD pair passing stops A and B. The output is then a list of those OD pairs and percentages, ready for the OD matrix calibration.

Algorithm 2: Finding OD Pairs (Tracks)

```
1 :  for all $i, j \in C,\ i \neq j$ do
2 :    perc = 0
3 :    for all $p \in paths(i, j)$ do
4 :      while goOn
5 :        take next $l \in p$
6 :        if $A$ is not found
7 :            if $l$ leads to stop $A$
8 :                if $TL^p(l)$ != $TL^p(l+1)$
9 :                    add $(A, TL^p(l+1))$
10 :               else
11 :                   goto next $p$
12 :               end
13 :           end
14 :       else // A is found, look for B
15 :           if $l$ leads to stop $B$
16 :               if $TL^p(l)$ != $TL^p(l+1)$
17 :                   add $(B, TL^p(l))$ , update goOn
18 :               else
19 :                   goto next $p$
20 :               end
21 :           elseif $TL^p(l)$ != $TL_A$
22 :               goto next $p$
23 :           end
24 :       end
25 :     endwhile
26 :     if $A, B$ found && $TL_A == TL_B$
27 :         perc = perc + $P_{ijp}$
28 :     end
29 :   end
30 :   if perc > 0
31 :       add$(i, j, \text{perc})$
32 :   end
33 : end
```

In the above algorithm all centroid pairs are checked, by looking at all links of the paths between the centroids. If the first stop has not yet been found the algorithm first looks for that stop. If the first stop has been found, the algorithm looks for the second stop, and also if the transit line has changed. Because if it did that means that the journey was interrupted, and the OD pair cannot be used for this Track.

If the algorithm finds one of the stops, and the transit lines remain the same, it means that the stop was not used in this path. Therefore immediately the next path is started. This of course assumes that the path will only visit a stop once, which is a reasonable assumption (if the path visits a stop twice it has a loop, and removing that loop would always result in a better path, therefore paths will not have a loop).

Algorithm 2 changes a little bit if it is applied to trains. In that case, on line 21 and on line 26, not the transit lines are compared, but instead the providers of the transit lines. For the rest the algorithm remains the same.

Algorithm 2 has a very large complexity, due to the fact that there is no easy bound on the number of paths in a network (this number can in theory become $n!$), or the number of links in a path (which could become a). Therefore in theory the complexity will be in the order of $O(c^2 \cdot n! \cdot a)$. However for OmniTRANS this bound can be reduced greatly. In most algorithms in OmniTRANS only the top ten paths are used, if there are more paths it is assumed that those will not add anything new, and are therefore skipped. Which means that the bound would reduce to $O(c^2 \cdot m \cdot a)$, with m the maximal number of paths used in the algorithm.

6.4 Comparing Flows at Stops

Now it is time to take a look at an algorithm to model the flows at a stop. For such an algorithm also more information from the route assignment is needed. At one stop it has to be known which OD pairs 'use' that stop. With 'using' it is meant that there is either an access-flow, an egress-flow or a transfer (either direct or by walking from/to another stop). If that is known, it is also known which OD pairs will have to be calibrated if the numbers in the OD matrix do not match with the values from the matrix H.

Because knowing which OD pairs use a stop is also part of what was needed for Algorithm 2, the same approach can be used for stop-modelling. Therefore it is assumed that for every centroid pair all paths between them are known (with a list of all traversed links), and which percentage of the total flow uses that path. Each link also 'knows' which transit line was used when the flow crossed that link, with a 0 if no transit line was used.

With the above information an algorithm similar to Algorithm 2 can be defined, though much simpler, as now only one stop will have to be checked. Again a lot is needed for the input: the set of centroid C, the set of all paths between any two centroids, $paths(i, j)$, the percentage of flow used by path p, P_{ijp}, and the transit line for every link l on path p, $TL^p(l)$. The algorithm checks if the centroid pairs use the given stop (stop A), and returns a list with those OD pairs, with a percentage and if the flow is a boarding or an alighting.

Algorithm 3: Finding OD Pairs (Stops)

```
1 : for all $i, j \in C,\ i \neq j$ do
2 :     $\text{perc}^{in} = 0,\ \text{perc}^{out} = 0$
3 :     for all $p \in paths(i, j)$ do
4 :         if $A \in p$
5 :             take $l^{in}, l^{out} \in p$, attached to A
6 :             if $TL^p(l^{in})\ != \ TL^p(l^{out})$
7 :                 if $TL^p(l^{in})\ != \ 0$
8 :                     $\text{perc}^{out} = \text{perc}^{out} + P_{ijp}$
9 :                 end
```

```
10 :                 if $TL^p(l^{out})$ != 0
11 :                     $\text{perc}^{in} = \text{perc}^{in} + P_{ijp}$
12 :                 end
13 :             else
14 :                 continue with next $p$
15 :             end
16 :         end
17 :     end
18 :     if $\text{perc}^{in} > 0$
19 :         add($i, j, \text{perc}^{in}$, 'board')
20 :     end
21 :     if $\text{perc}^{out} > 0$
22 :         add($i, j, \text{perc}^{out}$, 'alight')
23 :     end
24 : end
```

Because the above algorithm only needs to find one stop, it doesn't walk through the whole path like in Algorithm 2. Instead it checks if stop A appears in the path, and then takes the two links attached to the stop. If the transit lines attached to those links are different, the stop is used by the OD pair. There still is a difference between boarding and alighting. If the first transit line is 0 it is a boarding (access flow), if the second transit line is 0 it is an alighting (egress flow), and if both are non-zero both a boarding and alighting occurred (a transfer).

Again the above algorithm changes a little bit if it applied to trains. When looking at trains line 6 doesn't look at the transit lines, but instead at the transit line providers. The rest of the algorithm remains unchanged.

For the complexity of Algorithm 3, the complexity of checking if the stop A is in a path p has to be known. This complexity is dependant on what method is used, and how the path p is structured. For now the complexity is assumed to be $O(a)$, by looking through all links in the path p. With the for loop and the number of paths between two centroids, the total complexity of Algorithm 3 becomes $O(c^2 \cdot m \cdot a)$. This complexity is the same as for Algorithm 2, as basically the same steps are executed.

With three algorithms defined for the three approached defined in Chapter 5, in the next chapter these three approaches will be compared and discussed.

7. Different Options

With the different approaches from Chapter 5 and the algorithms from Chapter 6, three options to model the OV-Chipkaart data appear. In this chapter these three options will be treated and compared. In section 7.1 the different options will be treated. Section 7.2 then holds a small comparison between the options. And in section 7.3 something will be said about the influence of incomplete datasets.

7.1 The Options

The three approaches from Chapter 5, extended with the algorithms from Chapter 6, now give rise to three options. These options differ in how the information from the OV-Chipkaart data, contained in the matrix H, is taken into account with the OD matrix calibration. The options are described in detail below, to make them more formal and clear.

7.1.1 Option 1: Counts

The first option in modelling data from the OV-Chipkaart is to throw all information about lines, journeys and stops away, and just model the OV-Chipkaart data as counts. To get the number of passengers passing a specific link, all transit lines passing that link will have to be checked and added. For calculating a count Algorithm 1 can quite easily be used. After using it for every link between two stops in the network, a count is available for each of those links. These counts can then be used for OD matrix calibration, in the usual way as described in section 3.6.

Option 1 is primarily meant as a first impression, as it looses a lot of important information from the matrix H. The upside of Option 1 is that it doesn't take a lot of computation power, which means that it will very quickly get a result. The complexity of Option 1 is just the complexity of running Algorithm 1 once for every unique link on a transit line. Define q to be the maximum number of unique links between stops, and together with the complexity of Algorithm 1 $\big(O(e^2t)\big)$, the total complexity of Option 1 becomes $O(q \cdot e^2t)$.

7.1.2 Option 2: Tracks

The second option takes the journeys described in the matrix H into account, using them as a count on part of a transit journey, a Track. Every Track starts with a boarding or transfer, at the first stop, and ends with an alighting or transfer, at the second stop. Every Track therefore corresponds to exactly one cell in the matrix H. For Tracks it is important that the OD flow really uses the two stops, and not just passes it by.

The difficult part of Tracks is to find the actual OD pairs that use a Track, with both the boarding and the alighting. For finding these OD pairs Algorithm 2 can be used, if enough additional information about the route assignment is available. The information that is needed is every path between the two centroids, with at every link the transit line that was used, and the percentage of flow between the two centroids using that specific path. Algorithm 2 then has to be run once for every value in the matrix H, which is at most $s(s-1)$ times (if all stops are included in H).

Option 2 is perfect for big OD matrix calibrations, for when the original OD matrix is of poor quality or only generated by a gravity model. For small scale calibrations Option 2 will just be overkill. Algorithm 2 checks the paths between every centroid pair, which takes a lot of computational power. The complexity of Option 2 is determined by taking the complexity of Algorithm 2 (which is $O(c^2ma)$) $s(s-1)$ times, resulting in a total complexity of $O(s^2 \cdot c^2ma)$.

7.1.3 Option 3: Stops

The third option focusses on stops, and the flows entering and leaving them. To get the flows in and out of a stop first some aggregation is needed, as the rows and columns of matrix H will have to be summed. A row from H gives the number of departures from that stop, a column gives the number of arrivals at that stop. If these numbers are then compared with the results from the route assignment the flows using the stop can be calibrated.

With stop modelling it is also difficult to find the OD pairs that actually use a stop. An approach similar as in Option 2 is used, as described in Algorithm 3. For Algorithm 3 the same information is needed as in Option 2: every path between the two centroids, with at every link the transit line that was used, and the percentage of flow between the two centroids using that specific path. Algorithm 3 then has to be run for every stop in the network, to get the flows for each stop.

Option 3 is a good ‘fine-tuning’ approach. It will probably be quicker than Option 2, but it does also not use all information from the matrix H. The expectation is that Option 3 will be perfect for an OD matrix where only slight changes are necessary, like updating an OD matrix from a few years ago. The complexity of Option 3 is given by taking s times the complexity of Algorithm 3, $O(c^2ma)$, giving a total complexity of $O(h \cdot c^2ma)$.

7.2 Comparison

In the descriptions of the three options above already something has been said about when the options would be best used. This can be summarized in the following overview:

Option	**When?**	**Runtime***
Option 1	First impression	low
Option 2	Full calibration	high
Option 3	Fine-tuning	medium

(*) These are estimates, no actual implementation or experimentation has been used for these estimates.

One way to compare the different options of course is to look at their complexities. With the complexities of the three options however there is a problem. Because they have a lot of different variables, which are dependant on the model and the network, not much can be said to really compare them. Therefore instead of a theoretical comparison the complexities will be compared with a practical example: the 'Delft Basic' project, the standard example project in OmniTRANS.

In Delft the used variables take the following values:

variable:	c	a	s	t	e	m	q
value:	22	689	45	21	17	10	98

Which leads to the following complexities:

Option	**Complexity**	**Value in Delft**
Option 1:	$O(q \cdot e^2 t)$	594.762 steps
Option 2:	$O(s^2 \cdot c^2 ma)$	6.752.889.000 steps
Option 3:	$O(s \cdot c^2 ma)$	150.064.200 steps

The results from the Delft project are close to what was already predicted. Option 1 takes by far the lowest number of steps, and Option 3 the largest amount of steps. For the rest not much can be said, as nothing is known on how good they are with respect to OD matrix calibration. Therefore all three options will be implemented into OmniTRANS, to be able to also compare their runtime and the results. The process of this implementation can be found in the next chapter, Chapter 8.

7.3 Incomplete Datasets

With the options described and somewhat compared there is one aspect which still has to be treated, namely the quality of the data. This was briefly mentioned before, but now it will be looked at in more detail. The quality of the data will have an impact on the calibrations that are defined in the options above.

The data that is used by the above three options is data from the OV-Chipkaart. At this moment it is being introduced in the Netherlands, in most places it can be used but it is not the only valid method of payment. Which means that not all passengers will be travelling with an OV-Chipkaart, and therefore an OV-Chipkaart dataset generated at this moment will not be complete, it will only show part of all passengers on a transit line. This means that current OV-Chipkaart data cannot be used for normal OD matrix calibration.

7.3.1 Minimal Calibration

But what information does the matrix H give in that case? It shows passengers moving from stop to stop, but only a fraction of all passengers. Therefore it is known that there are **at least** that many passengers travelling on that transit line. Which means that they give a lower bound on the real amount of passengers, and therefore a lower bound on the amount of people travelling from one centroid to another.

The above can then be implemented as a "minimal calibration", which is quite similar to a 'normal' calibration. With a minimal calibration the OD matrix will only be calibrated if the restriction is larger than the real value. If the calibration would cause the OD matrix to decrease it will not be used in the calibration. This gives rise to the following definition of the solution methods:

(for the adapted OmniTRANS algorithm:)

$$g_{ij}^{new} = g_{ij}^{old} \cdot \left[\prod_{r \in R^+} \left(\frac{C_r}{\sum\limits_{i,j \in r} g_{ij}^{old} \cdot P_{ijr}} \right)^{\epsilon_r \cdot P_{ijr}} \right]^{\frac{1}{\sum_{r \in R^+} P_{ijr}}}$$

with R^+ defined as $R^+ = \left\{ r \in R \;\middle|\; C_r > \sum\limits_{i,j \in r} g_{ij}^{old} P_{ijr} \right\}$.

(and for the gradient descend method:)

$$\begin{aligned} \min_{\mathbf{g}} F(\mathbf{g}) &= \alpha \sum_{\substack{i,j \in I \\ d \in D}} \frac{1}{2} \left(g_{ij}^d - \hat{g}_{ij}^d \right)^2 + (1-\alpha) \sum_{r \in R^+} \frac{\epsilon_r}{2} \left(\sum_{\substack{i,j \in I, \\ d \in D_r}} P_{ijr}^d g_{ij}^d - C_r \right)^2 \\ \text{s.t.} \quad \mathbf{g} &\geqslant 0 \end{aligned}$$

with R^+ defined as $R^+ = \left\{ r \in R \;\middle|\; C_r > \sum\limits_{\substack{i,j \in I, \\ d \in D_r}} P_{ijr}^d g_{ij}^d \right\}$.

7.3.2 Improving OV-Chipkaart Data

Besides working with a minimal calibration, there is another option. If, for example from a survey, the penetration of the OV-Chipkaart under transit passengers is known, this percentage could be used to ‘complete’ the data. If for example the OV-Chipkaart data only shows 50% of all transit journeys, then the other 50% can be assumed to have the same journeys. In effect the OV-Chipkaart data can then be increased by a factor 2, giving a complete dataset.

By increasing the data from the OV-Chipkaart with a factor of course the quality of the data goes down. This comes from the fact that it is assumed that the other part of transit passengers has the same travel pattern as the passengers using an OV-Chipkaart. But that assumption doesn’t have to hold true. It could be very logical that OV-Chipkaart users are people who often travel with the public transit, and therefore have a whole different travel pattern.

But aside from the loss of quality the OV-Chipkaart data then can be used for a full calibration, giving more functionality. And as calibrating works with models the loss of quality can be quite acceptable, if it meant that a full calibration can be done. Therefore both approaches will be taken into account, for all three options.

The three options described in the above chapter will now be implemented, as described in the next chapter.

8. Implementation

This chapter treats the implementation of the three options from Chapter 7. The implementation will only be treated in general terms, specifics will be skipped. In section 8.1 the implementation of Option 1 is treated. Then in section 8.2 a problem with transit paths is discussed, which leads to a partial solution in section 8.3.

8.1 Creating Counts

The first prototype that was created was Option 1 (page 45). For Option 1 the Ov-Chipkaart data from matrix H is converted to a count on the link between every two sequent stops. The first implementation of Option 1 followed the steps from Algorithm 1. During the implementation two limitations of the algorithm appeared. Because of these limitations the algorithm had to be reworked, to be able to handle the situations causing these limitations.

The first limitation stemmed from the fact that it might happen that there are two complementing transit lines, like in the following example:

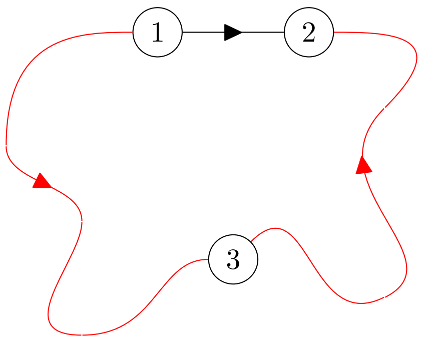


Figure 8.1: Two complementing transit lines

In the above example the two transit lines complement each other. A journey from stop 1 to stop 2 can either go directly, or through stop 3. The second option is of course very illogical, as it is much, much longer. Algorithm 1 however would find both journeys, and use both journeys for creating counts. Not only does this mean that a very illogical journey is created, but also that the passengers from 1 to 2 are counted twice.

The other limitation came from the possibility of using the algorithm for a train network. As a train network consists of multiple transit lines it becomes very difficult to determine which stops are predecessors or successors of one train stop. In the following picture this is easy to see:

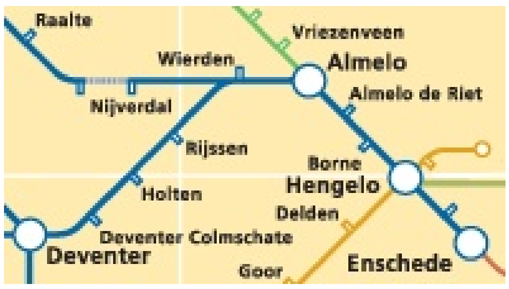


Figure 8.2: Train Network

In the above picture, if a journey is started from Enschede, then Almelo de Riet is a successor of Hengelo. And starting from Deventer, Almelo is a predecessor of Borne. But the journey Almelo-Almelo de Riet doesn't pass Borne-Hengelo at all. The reason that this happens is that successor list of a stop is different for every predecessor. And therefore it is impossible to use Algorithm 1.

To solve the above limitations another approach was produced. Instead of looking at both the predecessors and successors of a stop now only the successors of a stop are treated. From every stop all other reachable stops are checked, to see if there are passengers travelling between them. If there are, these passengers are added to all links between the two stops. If there are multiple routes between the two stops only the shortest one is taken into account, therefore prohibiting any double countings:

Algorithm 1b: Creating Counts (adapted)

```
1 : for all $f \in N_A^{\text{after}}$ do
2 :     take $p$ = shortest-path($A \to f$)
3 :     for all $l \in p$ do
4 :         $C(l) = C(l) + H(A, f)$
5 :     end
6 : end
```

In the above algorithm $C(l)$ is the count value of link l. There is still a difference between train journeys and non-train journeys, namely how the shortest path is found. For non-train journeys just the transit lines attached to the stop can be checked, and the shortest path is taken. For train journeys this is not possible, as multiple transit lines might have to be traversed before the second stop has been found. Therefore the following three step approach is used:

1. First check if both stops are on the same transit line.
2. Then compare the transit lines attached to both stops, to check if they overlap (have a stop in common).
3. Finally an adapted depth-first search algorithm is started, looking through the train network for a valid connection between the two stops, taking the shortest one.

Algorithm 1b and the two path finding approaches have been implemented as a job into OmniTRANS, without any more complications. Information about the results of the first prototype will be treated in the next chapter. Now first a look will be taken at the other two implementations.

8.2 Transit Paths

After the implementation of the first prototype a look was taken into the other two options. Almost immediately a large problem was found. In OmniTRANS the distinct paths of transit journeys cannot be saved. The only data that can be saved are access and egress flows, and screenline matrices for every link (for links that do not contain a count these are called “selected link matrices”). But a path, or even a way to match OD pairs to a transit line, are not available in OmniTRANS.

With the above problem the approach of Algorithms 2 and 3 cannot be used for an implementation in OmniTRANS. The reason that these paths are not available stems from the assignment algorithm that is used by OmniTRANS. Because calculating all paths and combinations of transit lines is far too complex the algorithm uses a series of cascading choice models. From every centroid there is a list of possible access candidates. Then from these access candidates passengers can board a transit line with a certain probability. At all stops containing more than one transit line there is a certain probability that passengers transfer to another transit line. And finally every centroid also has a list with egress candidates, completing the path. At the actual assignment the algorithm collapses all choice models, losing any information on where passengers came from or were travelling to.

After a lot of study and discussion with the designers, the conclusion was reached that there is no way to retrieve the used paths from the assignment algorithm. Therefore no direct solution exists how to solve this problem. Instead an approximation will be formulated, to find the most likely paths that were used.

8.3 Approximation

As mentioned before the only data that is available is the collection of access and egress flows, and for every link a screenline matrix. From this data a list with all OD pairs that use a stop (for Stop modelling) or use a Track (for Track modelling) has to be found. For Track modelling it has to be known if an OD pair boards at the first stop, alights at the second stop and follows the transit line for the whole journey. Therefore first a look is taken at just which OD pairs use a stop.

At a stop three types of flows can occur: an access flow (from a centroid), an egress flow (to a centroid) and a transfer (between two transit lines). From OmniTRANS for every centroid it is known which stops are used for access and egress flows, and to/from which transit lines these passengers board/alight. It is also known how many transfers from one transit line to another happened at every stop.

8.3.1 Access and Egress

With the given information it becomes possible to check which OD pairs use a stop. First the access and egress candidates are checked, because these are easier, as already the origin/destination of the OD pair is known. By checking the screenline matrix of the first/last link of the transit lines an impression can be gotten of how many passengers go to/from which destination/origin. Then, if the total amount of passengers is larger than the actual amount of passengers boarding/alighting, the total amount is lowered to be equal to the actual amount.

The above leads to Algorithm 4 (for access flows). In the algorithm a list called Access(A) contains the origin i, the unique starting links (l) of the transit lines attached to stop A (if two transit lines use the same link they are added together), and the value of passengers boarding the given transit line(s):

Algorithm 4: Checking Access Flows

```
 1 :  for all $i, l, value \in$ Access($A$) do
 2 :    for all $j \in C$ do
 3 :      if $S_l(i,j) > 0$
 4 :          add($i, j, S_l(i,j)$)
 5 :          $counter = counter + S_l(i,j) \cdot g_{ij}$
 6 :      end
 7 :    end
 8 :    if $counter > value$
 9 :        multiply all found OD pairs with $value/counter$
10 :    end
11 :  end
```

The same approach can of course be used for egress candidates. Only then the link l will be the last link before stop A on the given transit lines. Also the OD-pairs (j, i) will be checked, as the destination is already known.

8.3.2 Transfers

For the transfers more work is needed, as with transfers both the origin and destination of an OD pair have to be found. First the OD pairs found by looking at the access and egress flows are skipped, as they would cause double countings, giving rise to the set of remaining OD pairs I^-. As with access and egress flows the links on the transit lines are checked, but this time both the last link before and the first link after the stop are taken into account. The difference between these links is the amount of passengers that have boarded (if it is positive) or have alighted (if it is negative) at the stop. Together with the real amount of transfers it gives rise to the following algorithm:

Algorithm 5: Checking Transfer Flows

```
1 :  for all $l_{in}, l_{out}, value \in$ Transfer$(A)$ do
2 :     for all $(i,j) \in I^-$ do
3 :        if $S_{l_{out}}(i,j) - S_{l_{in}}(i,j) > 0$
4 :              add$(i, j, S_{l_{out}}(i,j) - S_{l_{in}}(i,j)$, 'board')
5 :              $counter = counter + (S_{l_{out}} - S_{l_{in}}) * g_{ij}$
6 :        elsif $S_{l_{in}}(i,j) - S_{l_{out}}(i,j) > 0$
7 :              add$(i, j, S_{l_{in}}(i,j) - S_{l_{out}}(i,j)$, 'alight')
8 :              $counter = counter - (S_{l_{in}} - S_{l_{out}}) * g_{ij}$
9 :        end
10 :    end
11 :    if $counter > value$
12 :          multiply all found OD pairs with $value/counter$
13 :    end
14 : end
```

Again the values are lowered if they are bigger than the actual number of transfers. In the above algorithm four situations can appear where the screenline matrices would indicate that no transfer happened, while it did:

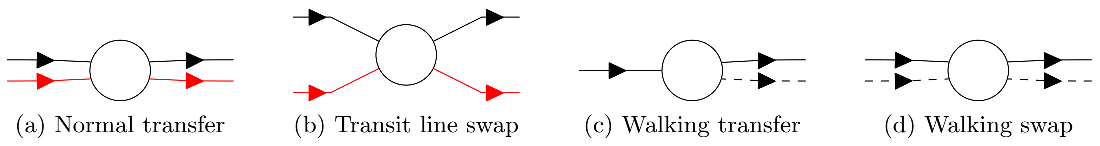

(a) Normal transfer (b) Transit line swap (c) Walking transfer (d) Walking swap

Figure 8.3: Transfer situations

In Figure 8.3a passengers transfer from one transit line to another, without this being registered in the screenline matrix. Figure 8.3b shows a situation where exactly the same amount of passengers transfer to the other transit line (from black to red and vice versa). Then in Figure 8.3c some passengers alight, and continue walking in the same direction. And in Figure 8.3d exactly the same amount of passengers board and alight at the stop.

The assumption is that situations 8.3b and 8.3d will almost never happen, as exactly the same amount of people have to swap, which is highly unlikely. For situations 8.3a and 8.3c one improvement is not to look at the last/first link on the transit line, but instead at the first link after the previous stop, and the last link before the next stop. This of course assumes that there are multiple links between stops, but in real world networks that is often the case. The improvement lies in the fact that the 'new' links are further away from the stop, and therefore there is a better chance that there is no overlap between another transit line or a walking journey, and therefore less chance of double countings.

There are also two situations where no transfer happened at the stop, but the screenline matrix would indicate that there was:

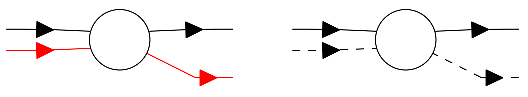

(a) Deviating transit lines (b) Deviating walk

Figure 8.4: More transfer situations

In Figure 8.4a the two transit lines first follow the same path, but after the stop divert from each other. And in Figure 8.4b the transit line overlaps with a walking journey, which again divert from each other after the stop. For both situations the improvement of looking at the other links, as described above, will mean that these situations will not occur often. But there is no real solution for these kinds of situations, everywhere where (transit)journeys overlap some miscounting will occur, as there is no link between transit lines and screenline matrices.

Finally, for Tracks there is one more thing to check. For the two stops attached to a Track all possible OD pairs can be checked, to see if they have both a boarding at the first stop, and an alighting at the second stop. What remains is to check if there is actually a flow on all links between the stops. If there is, the OD pair can be added to the list, if not the OD pair is skipped.

The above approaches have been implemented, conform with the descriptions of Chapter 7, as a Ruby job into OmniTRANS. Ruby is an object-oriented programming language, which has been ‘added’ to OmniTRANS. This was done so that users could write their own scripts, interfacing with OmniTRANS and using data and functionality that was already available inside OmniTRANS. In the next chapter a look will be given at some results from these prototypes.

9. Results

In this chapter the results of the three prototypes from Chapter 8 are treated. In section 9.1 first two testing networks are described. Then in section 9.2 several quality measures are treated, for comparing the three prototypes. In section 9.3 the different testing options are given, in section 9.4 the results from the Simple Network are given, and in section 9.5 the results from the Delft Network. Finally in section 9.6 some conclusions based on these results will be given.

9.1 The Test Networks

To compare the three different prototypes they have to be applied to a network. For this two networks were chosen, the simple network from [Smits, 2010] and the Delft project, the standard tutorial in OmniTRANS. These two networks will now first be described in some detail below.

9.1.1 Simple Network

The first network is based on the "Simple model" from [Smits, 2010], as shown in the following figure:

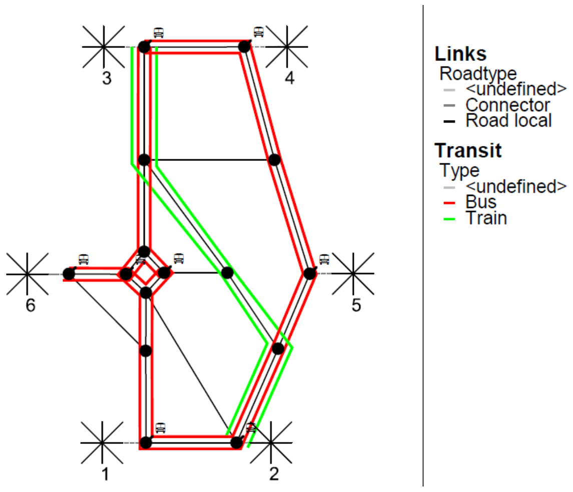


Figure 9.1: Simple Network

The above network consists of only six centroids, making it a perfect small-scale test. The different centroids are connected together by several bus lines and one train line, providing multiple routes between the centroids. The model only assigns and calculates public transport flows, as the other modes are not present in this network. The real OD matrix is given by:

	1	2	3	4	5	6
1	0	100	100	75	50	100
2	100	0	400	50	75	50
3	100	400	0	50	100	50
4	75	50	50	0	75	100
5	50	75	100	75	0	100
6	100	50	50	100	100	0

Table 9.1: OD Matrix for the Simple Network

For the starting OD matrix (the a priori matrix) the totals of the above matrix are used as the attractions/productions of the six centroids, after which a gravity model is run once to generate the starting matrix. From the real OD matrix it is easy to generate the matrix H, by taking the most logical routes through the network. The matrix H is given in Appendix C.

9.1.2 Delft

As mentioned before, the Delft network is the standard tutorial network in OmniTRANS. It consists of 22 centroids, connected by a large road network and 21 transit lines (figure 9.2). For the Delft network only some general socio-economic data was available, no real OD matrix containing the displacements in the city. Sadly also no real OV-Chipkaart data was available for the network.

Therefore both the 'real' OD matrix and the matrix H were generated by hand. The 'real' OD matrix was based on both the socio-economic data, the relative locations of the centroids and some general knowledge of Delft itself. The matrix H is generated by looking at logical routes through the network, with help from the shortest path algorithm in OmniTRANS. The 'real' OD matrix can be found in Appendix C, the matrix H was too large to print in this thesis.

9.2 The Quality Measurements

To be able to compare the different implementations the runtime is an important factor, as it indicates how much time the implementation uses. Another important factor is the results themselves, how much do they differ from each other and from the real OD matrix? To be able to compare the different results first some quality measurements have to be defined. These measurements are taken from [Smits, 2010], with one additional measurement (the absolute distance).

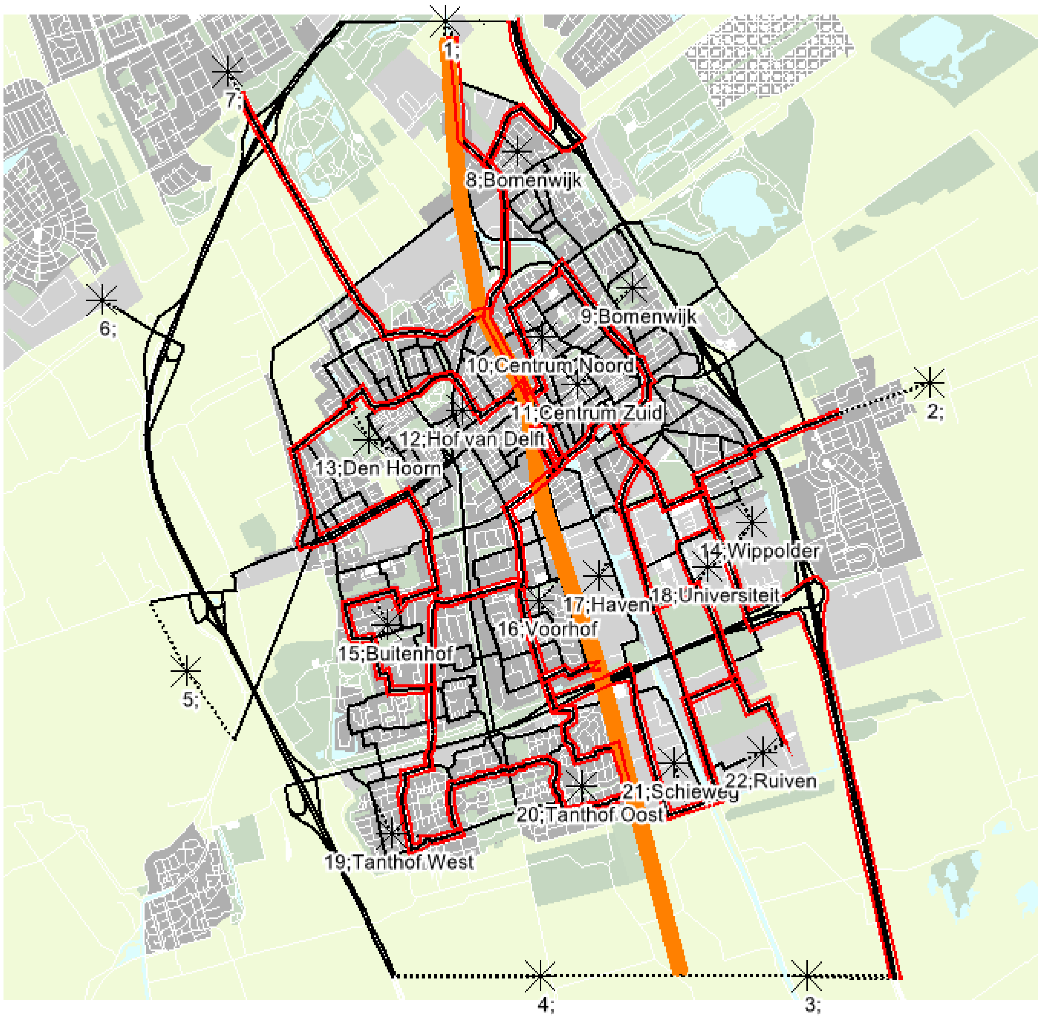


Figure 9.2: Delft Network

- **Coefficient of determination,** R^2

  The first quality measurement is based on statistics, and indicates the level of variance in the results. The coefficient of determination can both be applied to the variance between the starting and target OD matrix (**R^2-MAT**), and between the resulting OD matrix and the restrictions (**R^2-RES**). These measurements are given in the following formulas:

$$\mathbf{R^2\text{-}MAT} = 1 - \frac{\sum_{i,j}(g_{ij} - \hat{g}_{ij})^2}{\sum_{i,j}(g_{ij} - \bar{\hat{g}})^2}$$

$$\mathbf{R^2\text{-}RES} = 1 - \frac{\sum\limits_{r\in R}\left(\sum_{i,j} P_{ijr} g_{ij} - C_r\right)^2}{\sum\limits_{r\in R}(C_r - \bar{C})^2}$$

  here $\bar{\hat{g}} = \frac{\sum_{i,j}\hat{g}_{i,j}}{c(c-1)}$, $\bar{C} = \frac{\sum_{r\in R} C_r}{|R|}$ and with c the number of centroids.

  The coefficient of determination should be as close to 1 as possible, as then the variance is close to 0.

- **T-values, T_r and T_{te}**
  T-values are a quality measurement that is often used by Goudappel Coffeng. It is derived from the t-statistic, also from the area of statistics. The T-values are calculated for every restriction, checking the deviation between the final result for that restriction and the restriction value. The T-values are also calculated for the Trip ends separately (the T$_E$-values), to check the deviation between the starting matrix and the final OD matrix, after the calibration:

$$T_r = \ln\left(\frac{\left(\sum_{i,j} P_{ijr} g_{ij} - C_r\right)^2}{C_r}\right), \qquad T_{te} = \ln\left(\frac{\left(\sum_{i,j\in te} g_{ij} - \sum_{i,j\in te} \hat{g}_{ij}\right)^2}{\sum_{i,j\in te} \hat{g}_{ij}}\right)$$

  here $te \in E$, which is the set of Trip ends $\{(i,j) \mid i = z \in C\} \bigcup \{(i,j) \mid j = z \in C\}$, with C the set of all centroids.

  To use the T-values as a quality measurement the values have to be grouped together. This is done on their values, T-values that are below 3.5 are defined “good”, T-values below 4.5 are “acceptable” and values above 5.5 are “unacceptable”. In the Goudappel Coffeng guidelines it is given that for T_r’s at least 80% has to be good, at least 95% has to be acceptable, and no T_r’s can be unacceptable. For the T_{te}’s at least 90% has to be good, and no T_{te}’s can be acceptable or worse. Together this gives rise to the following six measures:

Name	Definition	Guideline
T3.5-RES	$= \frac{\lvert\{T_r \mid T_r \leqslant 3.5\}\rvert}{\lvert\{T_r\}\rvert} \cdot 100\%$	$\geqslant 80\%$
T4.5-RES	$= \frac{\lvert\{T_r \mid T_r \leqslant 4.5\}\rvert}{\lvert\{T_r\}\rvert} \cdot 100\%$	$\geqslant 95\%$
T5.5-RES	$= \frac{\lvert\{T_r \mid T_r \leqslant 5.5\}\rvert}{\lvert\{T_r\}\rvert} \cdot 100\%$	$= 100\%$
T3.5-TE	$= \frac{\lvert\{T_{te} \mid T_{te} \leqslant 3.5\}\rvert}{\lvert\{T_{te}\}\rvert} \cdot 100\%$	$\geqslant 90\%$
T4.5-TE	$= \frac{\lvert\{T_{te} \mid T_{te} \leqslant 4.5\}\rvert}{\lvert\{T_{te}\}\rvert} \cdot 100\%$	$= 100\%$
T5.5-TE	$= \frac{\lvert\{T_{te} \mid T_{te} \leqslant 5.5\}\rvert}{\lvert\{T_{te}\}\rvert} \cdot 100\%$	$= 100\%$

Table 9.2: Six T-value measurements

- **GEH statistics, GEH_r**
  The GEH statistic was originally invented by Geoffrey E. Havers, for transport planning. Though not a real statistics test, the GEH statistics work as an empirical formula. It is generally accepted as an acceptance criteria in the United Kingdom, where it has been used quite successfully for different traffic analyses. For every restriction the GEH statistic is calculated:

$$GEH_r = \sqrt{\frac{2\left(\sum_{i,j} P_{ijr} g_{ij} - C_r\right)^2}{\sum_{i,j} P_{ijr} g_{ij} + C_r}}$$

Again the GEH statistics are divided into groups. GEH statistics with a value of 5.0 or less are considered "good", GEH statistics with a value of 10.0 and lower are "acceptable", and if they are higher than 10.0 they are "unacceptable". According to English guidelines, at least 85% of all GEH statistics should be "good", and none should be "unacceptable". The GEH statistic gives rise the following two quality measurements:

Name	Definition	Guideline
GEH5	$= \frac{\lvert\{GEH_r \mid GEH_r \leqslant 5\}\rvert}{\lvert\{GEH_r\}\rvert} \cdot 100\%$	$\geqslant 85\%$
GEH10	$= \frac{\lvert\{GEH_r \mid GEH_r \leqslant 10\}\rvert}{\lvert\{GEH_r\}\rvert} \cdot 100\%$	$= 100\%$

Table 9.3: Two GEH statistic measurements

- **Absolute Distance,** AD
  The three measures described above look at the calibrated OD matrix and the restrictions defined in the model. But as these restrictions are partially generated by the prototypes themselves an additional quality measurement is needed, independent of these restrictions. Because for both networks the 'real' OD matrix is known, this measure can just be the absolute distance between the calibrated OD matrix and the 'real' OD matrix $\tilde{g}$:

  $$\mathbf{AD} = \sum_{i,j} |g_{ij} - \tilde{g}_{ij}|$$

  The AD measure is of course better when it is as low as possible, as then the calibrated OD matrix is very close to the 'real' OD matrix.

9.3 Testing Options

With the two test networks defined a look will be taken at what exactly will have to be tested. In Chapter 8 three prototypes were implemented, which have to be compared on their results and their runtime. But from these three of course another 'prototype' arises, namely using all three prototypes together. The expectation is of course that using all three prototypes will work better than the separate ones, but this is something that has to be tested first. Therefore there are four testing options, for easy references these will be called "COUNTS" (implementation from Option 1), "STOPS" (implementation of Option 3), "TRACKS" (implementation of Option 2) and "ALL".

Aside from the networks and the four options there is another aspect that has to be treated, namely the solution method. As mentioned earlier in this thesis two solution methods are used: the OmniTRANS adapted algorithm, and a Gradient descend method from [Smits, 2010]. As mentioned in [Smits, 2010] these two also rive rise to another solution method: a combination of both, by running them alternating. The three solution methods will be called "ADAPTED", "GRADIENT" and "COMBI".

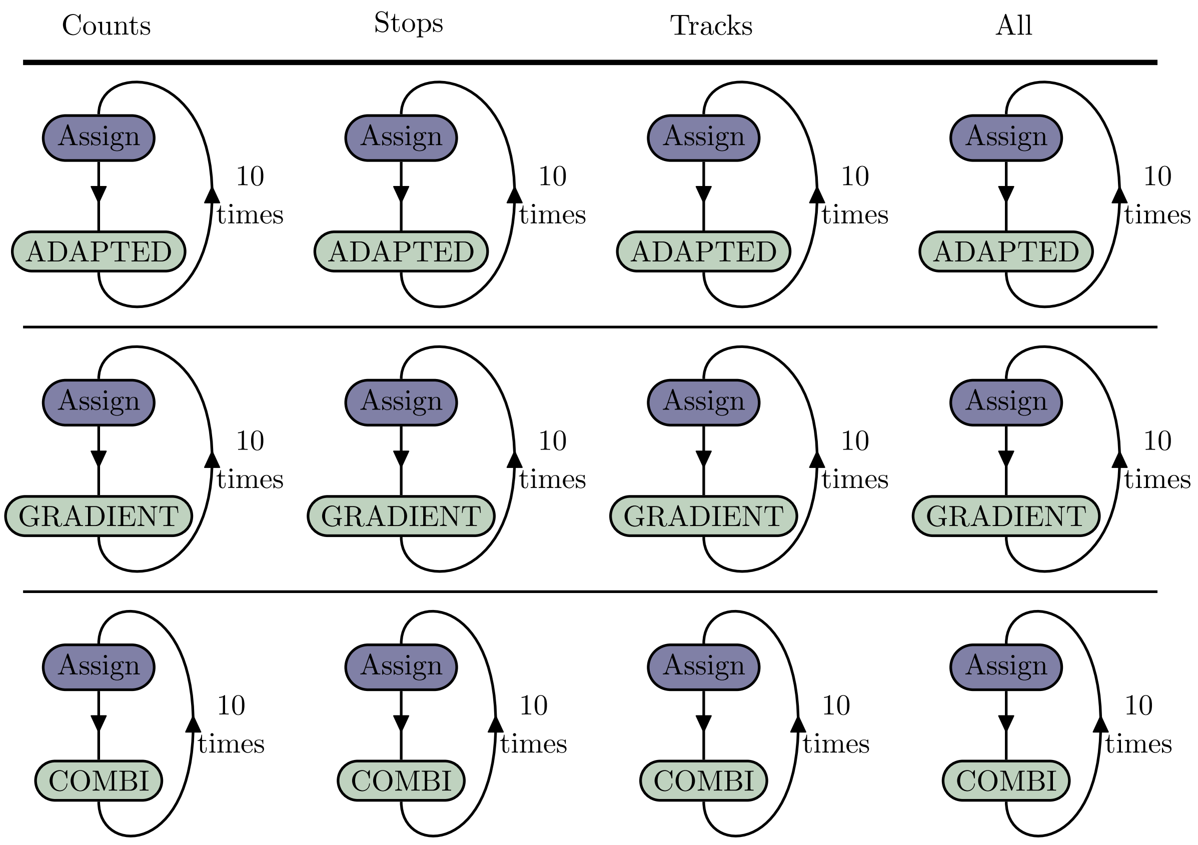


Figure 9.3: Three solution methods for each testing option

For the GRADIENT method, and therefore also for the COMBI method, first a suitable α has to be determined. As mentioned in [Smits, 2010] there is no clear way to see which α is good for a certain OD matrix calibration. Therefore for these results the optimal α is determined by running the solution method with different values of α, and choosing the best one. α is only calculated for GRADIENT, the same α is then used for the COMBI method. More about the calculation of α will be treated in Chapter 15.

9.4 Results - Simple Network

For the simple network the tests in Figure 9.3 are all run 10 times. Each calibration, i.e. one run of ADAPTED, GRADIENT or COMBI, has 20 iterations.

The results of the different tests is shown in Tables 9.4 and 9.5. If a test did not meet de guideline, it is shown in italics. First off, the run times of the different options behave quite like was predicted in Chapter 7. COUNTS has the lowest runtime, followed STOPS, TRACKS and finally ALL has the longest run time. The difference between the solution methods is also clear: ADAPTED is quicker than GRADIENT, and the runtime of COMBI therefore lies between that of ADAPTED and GRADIENT.

	Counts	Counts	Counts	Stops	Stops	Stops
	(adapted)	(gradient)	(combi)	(adapted)	(gradient)	(combi)
		$\alpha = 0.8$	$\alpha = 0.8$		$\alpha = 0.4$	$\alpha = 0.4$
R^2-MAT	0,99115	0,99868	0,98691	0,98936	0,99872	0,99085
R^2-RES	0,88036	0,92339	0,91125	0,86799	0,87390	0,87121
T3.5-RES	96,429	96,429	96,429	96,429	96,429	96,429
T4.5-RES	96,429	100,000	100,000	100,000	100,000	100,000
T5.5-RES	100,000	100,000	100,000	100,000	100,000	100,000
T3.5-TE	100,000	100,000	100,000	100,000	100,000	100,000
T4.5-TE	100,000	100,000	100,000	100,000	100,000	100,000
T5.5-TE	100,000	100,000	100,000	100,000	100,000	100,000
GEH5	89,286	96,429	96,429	96,429	96,429	96,429
GEH10	100,000	100,000	100,000	100,000	100,000	100,000
AD	2355,742	2094,982	1931,755	2431,246	2716,669	2535,225
Time	12,353	18,953	15,890	39,674	44,996	42,999

Table 9.4: Results for the Simple Network (1)

	Tracks	Tracks	Tracks	All	All	All
	(adapted)	(gradient)	(combi)	(adapted)	(gradient)	(combi)
		$\alpha = 0.75$	$\alpha = 0.75$		$\alpha = 0.8$	$\alpha = 0.8$
R^2-MAT	0,99999	0,99674	0,99733	0,94896	0,99878	0,96416
R^2-RES	0,96431	0,95659	0,96572	0,79221	0,86282	0,79145
T3.5-RES	93,750	90,625	96,875	*72,973*	*70,270*	*72,973*
T4.5-RES	100,000	100,000	100,000	*86,486*	*90,541*	*85,135*
T5.5-RES	100,000	100,000	100,000	*97,297*	*98,649*	*98,649*
T3.5-TE	100,000	100,000	100,000	100,000	100,000	100,000
T4.5-TE	100,000	100,000	100,000	100,000	100,000	100,000
T5.5-TE	100,000	100,000	100,000	100,000	100,000	100,000
GEH5	90,625	87,500	93,750	*67,568*	*68,919*	*70,270*
GEH10	100,000	100,000	100,000	*85,135*	*91,892*	*86,486*
AD	1055,197	727,136	990,135	994,937	1468,978	1260,787
Time	77,897	84,453	81,078	88,522	101,684	90,691

Table 9.5: Results for the Simple Network (2)

The above results show that the performance of the four options varies quite a bit, and therefore there is no clear ‘best’ option. Surprising is that ALL failed a lot of quality measurements, meaning that it is better to take COUNTS, STOPS and TRACKS separate than take them together. It could be possible that when more iterations are done that ALL might give better results, but that is something for another study.

Between the options COUNTS, STOPS and TRACKS, all of them achieved every quality measurements, independent of the solution method. The AD measurement is very high for COUNTS and STOPS, especially if the distance is compared to the real OD matrix. The sum of the entire OD matrix is 2950, which is only a little bit higher than the distance between the found OD matrix and the real OD matrix. Therefore the solutions generated by COUNTS and STOPS are in fact not very good representations of the real OD matrix. Only TRACKS comes close to the real OD matrix, but still with an absolute difference of about 25%.

9.5 Results - Delft

For the Delft network the same tests are run, as shown in Figure 9.3. The results from these tests are shown below, if a value is in italics it means that it did not meet the guidelines for that particularly test.

	Counts	Counts	Counts	Stops	Stops	Stops
	(adapted)	(gradient)	(combi)	(adapted)	(gradient)	(combi)
		$\alpha = 0.15$	$\alpha = 0.15$		$\alpha = 0.9$	$\alpha = 0.9$
R^2-MAT	0,99894	0,99921	0,97614	0,99914	0,99795	0,98224
R^2-RES	0,92574	0,96264	0,94585	0,62274	0,76691	0,62723
T3.5-RES	*57,273*	*65,455*	*63,636*	*24,444*	*23,333*	*32,222*
T4.5-RES	*71,818*	*77,273*	*77,273*	*45,556*	*37,778*	*48,889*
T5.5-RES	*89,091*	*88,182*	*86,364*	*65,556*	*56,667*	*62,222*
T3.5-TE	100,000	100,000	*81,081*	100,000	100,000	*72,973*
T4.5-TE	100,000	100,000	*94,595*	100,000	100,000	*83,784*
T5.5-TE	100,000	100,000	100,000	100,000	100,000	100,000
GEH5	*53,636*	*60,000*	*61,818*	*21,111*	*21,111*	*28,889*
GEH10	*75,455*	*80,000*	*79,091*	*47,778*	*38,889*	*50,000*
AD	63601,827	57548,447	60436,427	73651,518	67240,470	70380,689
Time	58,509	66,283	62,105	118,632	133,542	127,143

Table 9.6: Results for the Delft Network (1)

	Tracks	Tracks	Tracks	All	All	All
	(adapted)	(gradient)	(combi)	(adapted)	(gradient)	(combi)
		$\alpha = 0.15$	$\alpha = 0.15$		$\alpha = 0.95$	$\alpha = 0.95$
R^2-MAT	0,99857	0,99957	0,94707	0,99648	0,99834	0,95198
R^2-RES	0,90455	0,93351	0,91755	0,66858	0,83226	0,71958
T3.5-RES	*71,172*	*64,865*	*64,865*	*28,27*	*32,627*	*29,237*
T4.5-RES	*74,775*	*76,577*	*74,775*	*41,35*	*46,186*	*44,492*
T5.5-RES	*83,784*	*86,486*	*83,784*	*59,072*	*62,712*	*61,017*
T3.5-TE	100,000	100,000	91,892	100,000	100,000	*45,942*
T4.5-TE	100,000	100,000	*97,297*	100,000	100,000	*83,784*
T5.5-TE	100,000	100,000	100,000	100,000	100,000	*97,297*
GEH5	*66,667*	*59,459*	*62,162*	*24,473*	*27,119*	*24,576*
GEH10	*76,577*	*76,577*	*76,577*	*44,726*	*45,339*	*46,610*
AD	50048,061	49680,032	49532,808	61231,679	59752,468	58357,521
Time	886,485	893,860	891,403	926,744	973,000	935,237

Table 9.7: Results for the Delft Network (2)

The first thing that attracts attention is the fact that the T-RES and GEH tests all failed, none of the options ever succeeded. Also, the solution method COMBI managed to fail the T-TE tests, something which did not happen before.

The fact that so many tests have failed is perhaps easy to explain. The tests show how 'good' the OD matrix matches with the defined restrictions. But these restrictions are in fact defined by the prototypes. Therefore the fact that the tests are failed is an indication that the restrictions are not well-defined, making it impossible to match them to the OD matrix. This is also confirmed by the AD measure. The lowest distance between the found matrix and the real OD matrix is 49532,808. While the total sum of the real OD matrix is 44527, lower than the smallest absolute difference!

In comparing the four different options, again TRACKS comes out best. STOPS and ALL are the worst options, with COUNTS being in the middle. The balancing aspect is of course the runtime. Where COUNTS has a runtime of about one minute, TRACKS has a runtime of around 15 minutes! This increase is quite significant, especially if compared with STOPS, which uses part of the same algorithm. It therefore means that checking the different tracks and the transit lines takes a lot of computational power.

One possible explanation for the bad results in de Delft network can be seen in the route distribution in OmniTRANS. When the different results are compared, it seems that the generated OD matrices give rise to different journeys than with the real OD matrix. These route distributions are shown in Figures C.2, with the different colours showing how many people are travelling over that link. From a closer inspection it appears to indicate that the journeys generated by the OD matrices are shorter than planned, the calibrations give a preference to shorter journeys, as those are less 'restricted' than longer journeys, which pass more stops and Tracks. This is something that most definitely will have to be researched in more depth.

The fact that STOPS keeps getting bad results for both networks is perhaps explained by the fact that the used starting matrix is generated by a gravity model, while in Chapter 7 is was thought that STOPS would be good for small calibrations, just as a finishing touch. Therefore the above testing is not entirely fair. To give STOPS a better chance the tests for STOPS were run again, but this time with an OD matrix which differed only slightly from the real OD matrix.

Sadly this only improved the results of the tests by a bit, which is actually very bad. For the Test network the distance between the real OD matrix and the new starting matrix was 150, while the STOPS algorithm now gives an OD matrix with a distance of 2032,543. So even for the simple network STOPS managed to scramble the starting OD matrix to something neither resembling the real OD matrix or the starting matrix.

9.6 Conclusions

With the two networks tested, it is time to gather the results and derive some conclusions. The first remark that has to be made however is that the above results are influenced heavily by the implementation used in this thesis. As the implementation differs from the models on some point, not all results can be directly be connected to the models themselves. This also means that the models derived in Chapter 5 are sadly not proven or disproven with the implemented prototypes, only an indication of their merit can be given.

To see the effect of the three prototypes, it is interesting to check the starting matrix. For all of the above tests an OD matrix generated by a gravity model was used, which does not approximate the real OD matrix very well. This is obvious by looking at the difference between those two matrices. The absolute difference between the starting matrix and the real OD matrix is 2248,296 for the simple network, and the difference is 34360,335 for the Delft Network. If this is compared to the above results it shows that for the simple network the prototypes did get closer to the real OD matrix, but for the Delft network the OD matrix only got worse.

The conclusion therefore is that the prototypes for TRACKS and COUNTS do work, but not so good for large and complex networks. The cause of this is probably the approximations in the implementation, as the routes were not available. The expectation is therefore that with these routes available the prototypes will give better results. But at the moment it is not possible to test this. It is therefore an item on the list with recommendations, in Chapter 17.

The prototype STOPS and the combination of all prototypes, ALL, do sadly not work, even for small networks. The fact that ALL does not work might be dependant on the fact that STOPS does not work, therefore some more research will have to be done if COUNTS and TRACKS together work better than separately. Why STOPS does not work is also something that should be examined more closely. One possibility is that the aggregation is too much, that the calibrations done by the prototype are just too general to give good results. Another possibility is that the routes in OmniTRANS are just a little bit off from the model, which might cause the calibrations to get diverted.

Part II

Mobile Phones

10. Modelling Mobile Phone Data

With public transport treated in Part I it is now time to take a look at mobile phones. In this chapter a first step will be taken in the modelling of this area. In section 10.1 first something will be said about mobile phone data in the literature. Then in section 10.2 the structure of the data is treated, for both GSM and GPS datasets. In section 10.3 the initial modelling of the problem is described, and in sections 10.4 and 10.5 some information will be given on converting coordinates.

10.1 Other Literature

In the Literature Research, Chapter 2, some relevant literature was found concerning mobile phone data. There have been several attempts to use mobile phone datasets for a variety of studies and calibrations, with varying results. Not many studies used the data for actual OD matrix calibrations, but there were some publications mentioning it.

For GSM datasets three general approaches were found in the literature. The first approach only used GSM data that could easily be converted to precise locations, and therefore used the data with a GPS approach (as described below). The second approach received GSM data from a roadside monitor, which is not feasible for large studies. And the third approach used the GSM data by defining areas the mobile phone passed by, and then generating all possible routes, choosing the best one. This approach is also the one that will be used in this thesis.

For GPS data more research was available, because of the good accuracy of generated measurements by GPS locators. Therefore it is relatively easy to find the route that a GPS locator took. Most studies were done with datasets generated by car navigation systems or specially placed GPS locators. However the approaches used for those datasets can be easily generalized to include datasets from mobile phones.

10.2 Data Structure

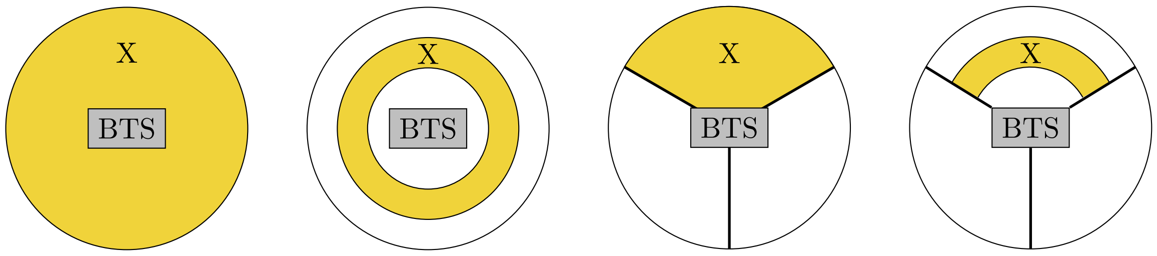
The data generated by GSM and GPS devices has a different structure than data generated by the OV-Chipkaart. Where OV-Chipkaart data has information on the start and end of a trip, and not the route, GSM and GPS data has only the route, but often no clear start or end. There is also a lot of difference between GSM data and GPS data, and therefore both types will be treated separately.

10.2.1 GSM

Aside from being different from GPS data, GSM datasets also differ from each other. This difference has several reasons, as some transceivers do not have the same capacities to store and send the same data, or due to privacy reasons some providers might delete more information than others. There is also a difference in which country the data is from, as different countries have different rules concerning privacy and the minimum capabilities of transceivers.

In the best case scenario a GSM dataset consists of regular measurements of the nearest base transceiver stations to the phone, with the response times (time it takes for a signal to go reach the station) to all stations. With those measurements first the distance between the different stations and the person can be calculated (based on the response time), then with those distances the location of the person in question can be found. If all that information is available the measurements of the location are so good that they can be used as GPS coordinates (see section 10.2.2).

But the above best scenario is often not the case. It is far more likely that the data is less detailed. For instance often only the nearest base transceiver records data about a mobile phone, other transceivers in the neighbourhood don't make a connection and therefore do not record anything. In that case an exact location cannot be found, only an area where the mobile phone can be found. Depending on the fact if the distance and/or the direction of the cell is known, the area is smaller and better defined:



(a) The whole area (b) Distance is known (c) Direction is known (d) Both distance and direction are known

Figure 10.1: Location area of a person X, transmitting to the base transceiver station BTS.

It can also happen that measurements are not saved regularly. This means that large gaps between different measurements can exists, sometimes leading up to hours at a time. Such gaps make it difficult to find the exact route a person took, especially if a large distance was travelled in that gap. If there are large gaps between measurements it can also mean that the journey actually consists of two smaller journeys, or just that the person is in an area with limited coverage by the provider. All these things have to be taken into account when modelling GSM data.

In general GSM data will always consist of the following: a user ID (to connect different measurements, but hide any personal information), the coordinates of the transceiver, the time stamp (often with a time zone and date) and the range of the transceiver. In specific cases GSM data can then be expanded with information about the response time of the phone and/or the cell that was used (giving the general direction of the mobile phone).

10.2.2 GPS

Data from a GPS provider is formatted as a large table, containing the different measurements from that provider. Each row in the table will at least hold the following information: a user ID, coordinates of the measurement and a time stamp (expanded with a time zone and date). Sometimes other information is added too, for example the altitude, but for the general case this information is assumed absent.

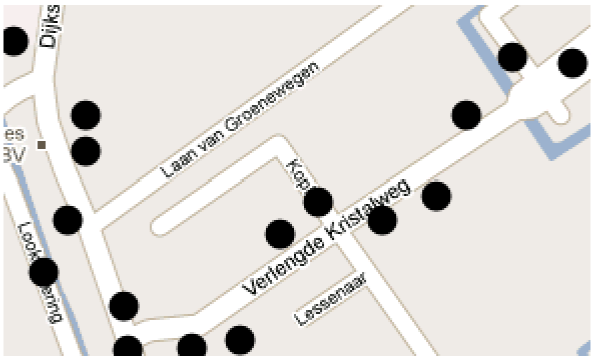


Figure 10.2: An example of GPS coordinates

GPS datasets can still differ from each other, by the format in which the coordinates are given. The standard format is of course longitude and latitude, but not all locators use that format. Other formats that can be used are for example WGS84 or NAD27. These formats (or datums as they are often called) differ in the modelling of the shape of the earth. The earth is not a sphere, but has an irregular shape. As there are different ways to approximate the shape of the earth, there are also different formats for writing down coordinates on the surface of the earth.

10.3 Initial Modelling

The modelling of GSM and GPS data has a bit more complexity than modelling data from the OV-Chipkaart, but on the other hand this also means that there are less options to consider. Data from GPS consists of a series of points, data from GSM consists of a series of areas. Both cannot be used as they are, so they have to be converted to whole journeys.

For the modelling of the mobile phone data a rough action plan can be created:

GSM	**GPS**
Process raw data	Process raw data
Find coordinates (if possible)	Convert coordinates
Generate Routes	Generate Routes
Model modal split	Model modal split
Calibrate OD Matrix	Calibrate OD Matrix

- **Process Raw Data**
  The raw data that is collected from a GSM or GPS provider first has to be processed. This includes checking the data for strange numbers (large jumps in time/distance), putting the data in a usable format and (if that was not yet done) remove any personal information. As this processing is mainly something that needs no modelling it is assumed that the user takes care of it, delivering the data in a processed structure.

- **Find Coordinates**
  For the best case scenario with GSM data (see section 10.2.1), the GSM data can be converted to exact locations. The conversion of GSM data to coordinates works by finding the distance to several transceivers, and then triangulating the location of the mobile phone. This conversion is independent of time, and therefore has to be done for every time stamp. More about the conversion of coordinates will be discussed in section 10.4.

- **Convert Coordinates**
  The coordinates that are given in GPS datasets are encoded in a specific format. Sometimes the format is the well-known longitude and latitude, other times it can be a datum like WGS84 or NAD27. But no matter the format these coordinates have to be converted to meters, as otherwise they cannot be used for calculating speeds. For more information about this conversion see section 10.5.

- **Generate Routes**
  With the (converted) coordinates or areas from the above steps the actual route that a mobile phone traversed can be derived. It depends on whether the data is structured as points or as areas which algorithm and approach has to be used. The process of finding these routes is called Route Mapping, and will be treated in the next chapter, Chapter 11.

- **Modal Split**
  From the Route Mapping for every user ID a route is given, showing how the mobile phone traversed through the network. But what is not immediately apparent, is the actual mode of that journey. The person carrying the mobile phone could be travelling in his own car, or he/she could be riding inside a bus. Therefore it is important to try and split the routes over the different modes. Otherwise the car OD matrix could be calibrated with people travelling by train, and that is something that is not supposed to happen. More about the modal split will be explained in Chapter 12.

- **Calibrate OD Matrix**
  And then finally the routes have to be used for the calibration of the OD matrix, based on the mode from the modal split. How the OD matrix calibration is accomplished will be treated in Chapter 12.

10.4 Finding coordinates

With detailed information about a mobile phone and the surrounding transceiver stations it becomes possible to find an exact location of that specific mobile phone. For this the following is assumed known: the user ID, time stamp, the response time to at least three base transceiver stations and the exact location of those stations.

First the response times have to be converted to distance. Radio waves, like the ones used for GSM networks, travel at the speed of light, $\approx 3,0 \cdot 10^8$m/s. If the response time is in seconds it can easily be converted to a distance: $distance = response\ time \cdot 3,0 \cdot 10^8$.

With the distances to the different transceivers and the location of those transceivers known it becomes simple to find the location of the mobile phone. This is easiest explained with a picture:

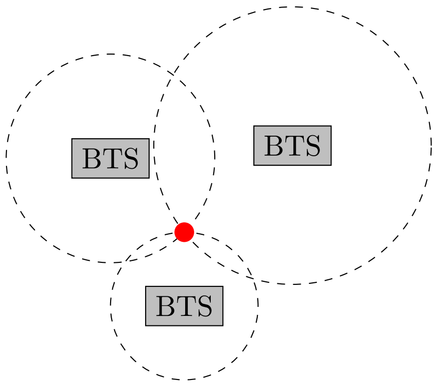


Figure 10.3: Triangulation of a mobile phone

As can be seen in the above figure, the mobile phone is a certain distance away from the base transceiver stations (BTS's). This distance is shown with the circles around each station. Then the location of the mobile phone is exactly where the three circles meet, shown with the red dot.

When all measurements of a user ID have been converted to coordinates the dataset becomes similar to GPS data, and therefore can also be used as such.

10.5 Converting the coordinates

As mentioned before the coordinates from GPS data have to be converted to meters, to be able to calculate speeds between different points (the speed is the length divided by the time). The speeds are needed later on, for finding optimal routes and for the modal split. Most of the data however is not given in meters, but in longitude and latitude, in degrees. Longitude and latitude do not have a straight relation with a normal grid, as can be seen in the following picture:

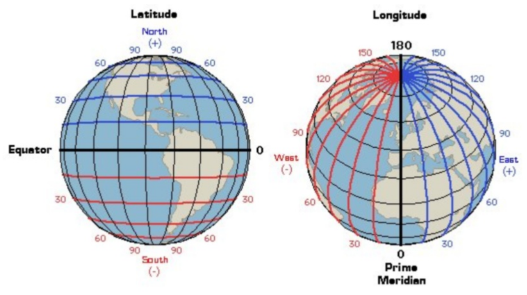


Figure 10.4: Latitude and Longitude

Longitude and latitude lines follow the curve of the earth, and therefore the longitude is slightly skewed. The conversion is quite simple:

$$x = \textbf{latitude} \cdot 111,2 \cdot 10^3$$
$$y = \textbf{longitude} \cdot 111,2 \cdot 10^3 \cdot \cos(\textbf{latitude})$$

In these formulas the longitude and latitude are in radials. The conversion from decimal degrees to radials is $long^{\text{rad}} = long^{\text{deg}} \cdot \frac{\pi}{180}$. It can also happen that the longitude and latitude are written in degrees, minutes and seconds (dd:mm:ss). The conversion from that to decimal degrees is $long^{\text{dec. degrees}} = dd + 60 \cdot mm + 3600 \cdot ss$.

Now it can still happen that the dataset is provided in another format. Then the dataset has to be first converted to longitude and latitude, and then to meters. Information about how to convert other formats (like WGS84 and NAD27) can be found on [Dutch, 2010].

11. Route Mapping

In this chapter the process of generating routes will be discussed. Route mapping is the process of turning the measurements (GSM areas or GPS points) into routes over the network. For this two algorithms will be discussed below. First in section 11.1 a new algorithm will be formulated for GSM areas. Then in section 11.2 an existing algorithm for GPS points will be discussed.

11.1 GSM Algorithm

As mentioned in the Literature Research, Chapter 2, no real algorithms have been found for mapping routes from a GSM dataset. Therefore a new algorithm will have to be created, to facilitate this. There was one paper, [van der Zijpp, 2005], which provided a general approach on how to map GSM data to a set of possible routes. This general approach will be used as basis for the GSM described below.

The approach from [van der Zijpp, 2005] works by finding paths that pass through all the areas from the GSM dataset, and then find the best path by adding some constraints. In the following algorithm this approach is changed a bit, as not all paths are taken into account, specifically only the shortest paths between the areas are taken into account.

11.1.1 Initialisation

The first step in the algorithm is to convert the measurements from the GSM data to actual areas. As shown in Figure 10.1 an area can take four different forms: a circle (only the transceiver and range are known), a ring (the distance to the transceiver is known), a sector (the direction relative to the transceiver is known) or part of a sector (if both the direction and distance are known).

The areas from the first step are then grouped into series. One series will contain all the areas of one user ID, ordered by time stamp. Therefore that series will show one complete journey, made by that particular user ID. Later on it might happen that such an journey might be split into two or more journeys, but that is something that will be explained in section 11.1.4.

In a series of areas it can happen that two sequent areas are the same. In that case the second are will be deleted, as it will not add any additional information. It can happen when a mobile phone is standing still, or when it is travelling very slowly.

Furthermore it might happen that measurements switch between two transceivers. Switching might happen if the mobile phone makes a u-turn (returns in the same direction), or if the phone is moving through the overlap between the two transceivers. In the second case using both measurements would make for very strange journeys. Therefore, if it happens in a small time interval (say 1 minute), the area will be deleted.

For example: if the mobile phone moves from area A to area B, and then back to area A, area B and the second area A will be removed from the series. There is of course a better way to handle this, more about that will be said in the section 11.1.4.

11.1.2 The Algorithm

The following algorithm will be run once for every series generated in the Initialisation. The input is therefore the series of areas, with size m. The algorithm then generates a lot of routes between these areas, and finally returns the best path. The best path is defined by a series of sequent links, and is therefore the output of the algorithm:

GSM Algorithm

```
1 :  $N_1 = \{p \mid p \in \text{area } 1\}$
2 :  for $i = 2 .. m$ do
3 :    $N_i = \{p \mid p \in \text{area } i\}$
4 :    for all $p \in N_{i-1}, q \in N_i$ do
5 :        find shortest path between $p$ and $q$
6 :    end
7 :    keep all unique paths
8 :  end
9 :  take best path
```

The algorithm also has two filters implemented, to make sure that only complete paths are found and to decrease the runtime. These filters are visualised in the following example:

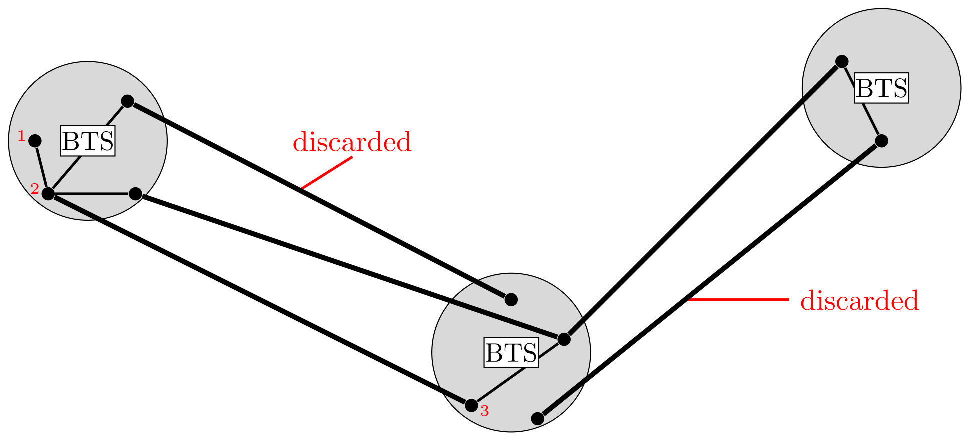


Figure 11.1: Two Filters for the GSM Algorithm

The first filter works by taking only unique paths. In the above example there are (amongst others) two paths: one going from 1 to 3, and one going from 2 to 3. But as the path from 2 to 3 is contained entirely in the path from 1 to 3, it is useless and therefore not saved separately. The second filter works by skipping paths that are not complete. In the above example there are two of such paths, with the comment that they will be discarded. This situation will not happen very often, as it can only occur if the path travels a disjunct part of the network.

After the route has been found, there is still one more step that has to be taken: an origin and destination have to be found. This is solved in a very simple way: just take the nearest centroid to the first node as the origin, and the nearest centroid to the last node as the destination. And then the algorithm can continue with the next route.

11.1.3 Comments

The algorithm defined in the above section is of course a very simple one. Instead of looking at 'good' paths, it only takes the shortest paths into account. And as it is possible that there are a lot of nodes in each area, it is also possible that a lot of candidates paths are found, from which only the 'best' path is chosen. Therefore there are two comments that have to be made.

First there is the complexity. Not much can be said about the overall complexity, as it depends heavily on the used algorithm to find the shortest paths and on the number and size of the GSM areas. If for example the Dijkstra algorithm would be used the complexity of the shortest path would be $O(n^2)$, with n the number of nodes in the network. An (very large) upper bound on the number of shortest paths that have to be calculated would be $O(n^2 \cdot n^{m-1}) = O(n^m)$, with m the number of areas in the dataset. But both complexities will almost never be reached, as the areas will never contain all nodes, and for calculating the shortest path not the entire network will have to be mapped.

The other comment is on the definition of the 'best' path. The quality of a path can be based on different cost-functions. For instance: the total distance, the total travel time over the path, or any other cost-function. It is also possible to mix cost-functions, though it will increase the complexity of the algorithm. Which cost-function is used will be a choice for the user, as it will be dependant on the wishes of the user which path will be considered 'best'.

11.1.4 Improvements

As mentioned before the GSM algorithm is quite simple. This means that there are quite a few improvements possible, to improve the algorithm. In this section a few easy improvements will be given, which will be taken into account with the actual implementation:

- The first improvement is to also look at the speeds over the paths. If the speed of a mobile phone is too high or low to be able to use a certain link in the shortest path, then that path should be skipped.
- Continuing with the above improvement, instead of looking at the costs (or perhaps in addition to), it might be interesting to look at the path that based on the speeds over the links resembles the 'real' situation the most.
- Something which should also be checked is the road type. A path changing from train tracks to a road, and then back again is of course faulty. That kind of situations should be checked for and taken into account with the route mapping.
- The act of deleting double measurements (going from centroid A to B and back to A) can also be handled in another way. If the two transitions happen quickly enough after each other it might be better just to replace them with one area, the overlap between areas A and B. That way the information that the mobile phone was close to both transceivers is not lost.
- As mentioned before it might be an option to take long journeys, that have a large break in the middle, and split them into two or more journeys. How and when this splitting must be done is something which should be researched further, during the implementation of the prototype.

The improvements listed above will be considered in the implementation of the prototype. More about the implementation of the prototype can be found in Chapter 13.

11.2 GPS Algorithm

With an algorithm defined for GSM datasets, now a look will be taken at GPS datasets. As mentioned before there has been more literature about GPS points, as these are much easier to model. The algorithm that will be used in this thesis is the algorithm from [Fafieanie, 2009]. The reason for choosing this algorithm is because it is already implemented, with the ability to interface with OmniTRANS.

The algorithm described in [Fafieanie, 2009] is part of a larger implementation, designed to generate a route set based on a GPS dataset. The algorithm uses information from a group of car navigation systems as input. And although in this thesis information from mobile phones is used, no large adjustments are necessary, as there is no real difference between car navigation coordinates and mobile phone coordinates.

The GPS algorithm works by first mapping the coordinates from the GPS dataset to a route, and then using these routes to generate a set of most likely candidates for the route set in OmniTRANS. For the purpose of finding routes from GPS data only the first part of the algorithm is of interest, and therefore only that part will be described in the next section.

11.2.1 The Algorithm

For the GPS algorithm first the following variables are needed:

G(V,E)	Graph describing the network, (with V the set of nodes and E the set of links)
AB	The link between nodes A and B
$d_e(A, B)$	The Euclidean distance between two points A and B
Q_i	A GPS point, consisting of a pair of coordinates (x_i, y_i)
$Q_{1..T}$	Set of points given by the GPS data
Q^t	The projection of Q on the link AB
$p\{E_1..E_p\}$	The path along links E_1 to E_p (with E_i, E_{i+1} sequent).

The projection Q^t of the GPS point Q on the link AB can be found as follows:

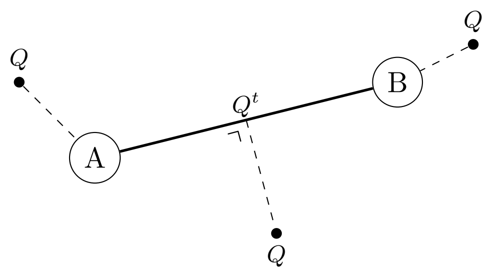


Figure 11.2: Three possibilities for Q^t

As is shown in the above picture Q^t is the point on the link AB that is closest to the point Q, which can be found by taking QQ^t perpendicular to AB. If Q falls outside the scope of AB, then Q^t becomes either A or B, whichever is closest. This immediately gives the following definition for the distance between Q and the link AB:

$$d(Q, AB) = \begin{cases} d_e(Q, Q^t) & \textit{if } Q^t \in [AB] \\ \min\{d_e(Q, A), d_e(Q, B)\} & \textit{elsewhere} \end{cases}$$

With the above definition of distance, it becomes possible to find the closest link to a GPS point. If it is done for several GPS points a series of links will be found, and as GPS points are quite close to each other, these links will be sequent. Continuing with this approach a path p is generated, containing the links attached to every point Q_i. As sometimes multiple paths might appear, a measure has to be defined to be able to compare different paths. This is done with the following 'score' F_p:

$$F_p = \sum_{j=1}^{|p|} \sum_i d(Q_i, E_j) \cdot \delta_{ij}$$

Here $\delta_{ij} = 1$ if Q_i is matched to E_j, 0 otherwise.

The precise steps of the GPS algorithm are shown in the following flow chart:

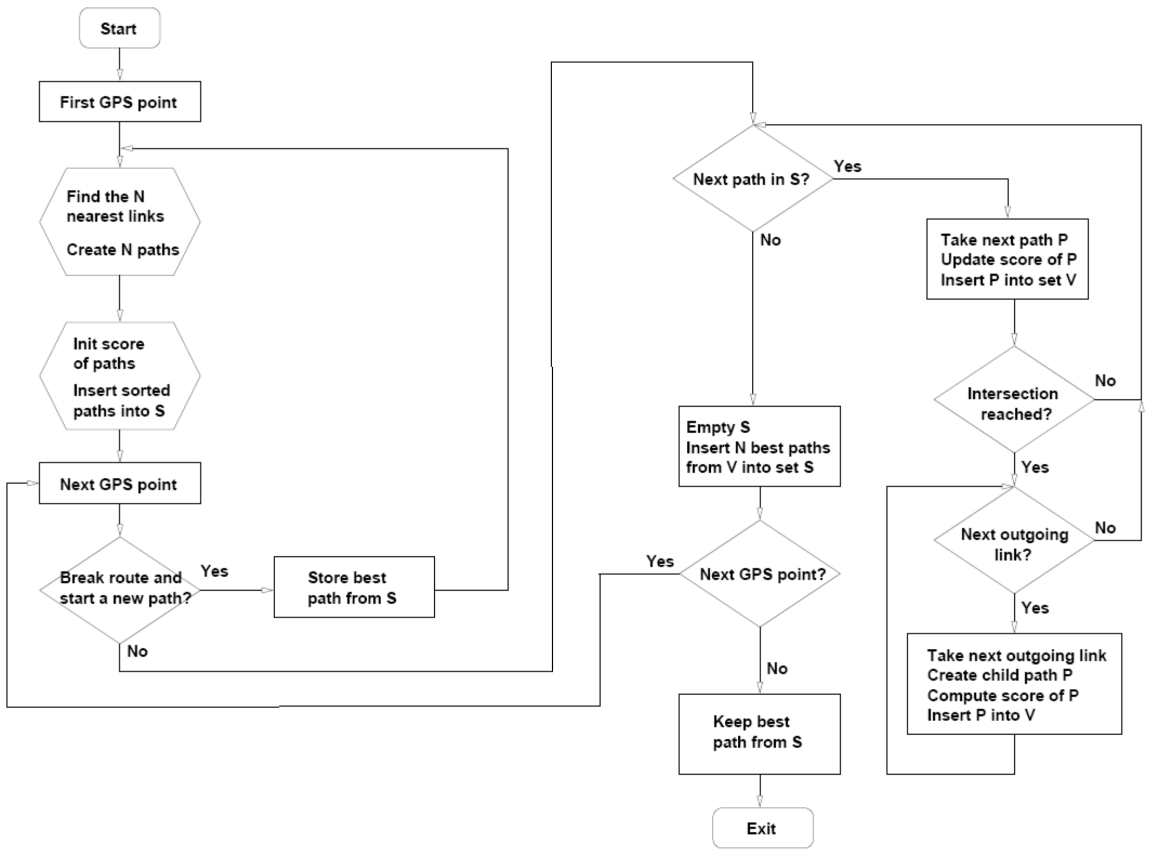


Figure 11.3: Flow Chart for the GPS Algorithm (from [Fafieanie, 2009]).

With the above GPS algorithm a set of paths is found, each path belonging to a user ID and following the route as given by the GPS coordinates of that user ID. As these paths lack an actual starting point and endpoint, these are then generated for every path. This is done by taking the closest centroid to the first link and the closest centroid to the final link, just like in the GSM algorithm.

For more information about the GPS algorithm and its implementation see [Fafieanie, 2009].

11.2.2 Modifications

There are a few adjustments that have to be made to the GPS algorithm to get it fully functional for the GPS datasets that will be used in this thesis. The GPS algorithm was based on GPS signals generated by a car navigation system. This means that only the car network had to be modelled, and certain assumptions could be made. The datasets in this thesis are more general, and therefore these assumptions will have to be generalised.

- The graph in [Fafieanie, 2009] only represented the car network. For the general case also train tracks, bus-only roads and even bicycle paths have to be added. Otherwise if a mobile phone travels by bicycle for example, it will be modelled to other roads, which can lead to some confusion (a bicycle riding on a highway for example).

- With the introduction of train tracks a new check might have to be implemented, to correct paths that switch from train tracks to a road, and back again. It might be that it will not happen within the algorithm, but it is something that should at least be looked at.

- GPS systems are dependant on the signals from satellites in orbit. This means that if a mobile phone enters a metro or subway station, the signal is lost. And if the mobile phone reappears somewhere else, it is modelled as a new journey. But it is quite clear that it is one journey, part of which is traversed with the metro. Some kind of check for these kind of situations will have to be implemented, to improve the algorithm.

- And the last modification is to remove the filters that were implemented in the algorithm. As the original objective of the algorithm was to find a set of likely candidates for a route generation set, several filters were added to remove undesired paths. But in the situation of this thesis there are no undesired paths, and therefore these filters have to be skipped. It will also mean that perhaps too many routes will be generated, but that is again something that will have to be checked during the implementation.

The GPS algorithm already has an implementation (in Ruby), therefore only the above modifications above will have to be implemented. This implementation will be further discussed in Chapter 13. In the next chapter first the routes, generated by the above algorithms (both GSM and GPS), will be processed so that they can be used for OD matrix calibration.

12. Route Processing

In the last chapter two algorithms were given to convert the GSM and GPS datasets to routes. These routes cannot be used as they are, first they have to be processed. Processing the routes consists of two parts. In section 12.1 the routes are split based on their mode, as public transport journeys cannot be compared with car journeys. Then in section 12.2 the routes are adjusted, to convert the number of phones to the number of cars along a route. And finally in section 12.3 the actual OD matrix calibration will be given.

12.1 Modal Split

The routes that were generated by the algorithms in the last chapter show the number of displacements between two centroids. But sadly these displacements are not the same kind of displacements in an OD matrix. A person with a mobile phone can be sitting inside a car or he/she can be riding inside a train, the signal and therefore the measurements are the same. And thus a modal split has to be executed.

It can be quite difficult to find the exact transportation mode of a mobile phone, but for certain situations there are easy 'clues' that indicate the mode. These clues can be used to immediately find the mode. An overview of these clues is given below:

- If the speeds over **all** links is below 10 km/h, the mode is Walk.
- If the speeds over **all** links is below 30 km/h (and above 10km/h), the mode is Bicycle.
- If the route follows train tracks, the mode is Public transport.
- If the route follows a bus-only road, the mode is Public transport.
- If speeds over **some** (non train track) links is above 100 km/h, the mode is Car.

All routes that are left over after the above modal assignment still have to be assigned to a mode. It is known that these routes are travelling through the normal road network, therefore the mobile phones are travelling either by car or by tram/bus (public transport). But how to split these remaining routes is quite difficult.

One possibility is to look at the exact route, to see if it passes by the stops belonging to a certain transit line. But if a transit line follows the path that a car also would have taken, it is still not certain in which mode the mobile phone travelled. Another check is to look at the times, if they resemble the timetable of a bus or tram line.

Sadly there is no foolproof way to split the routes generated by mobile phones, because there is no way to exactly determine the used mode. There will always exist situations where the mode is impossible to deduce from the available data. Therefore it will be up to the user to decide what to do with the remaining routes, and how to split them between car and public transport.

12.2 Adjusting the Routes

With the modal split complete there is one problem remaining. It was briefly mentioned before, but it is quite an important problem: the amount of mobile phones travelling on a link is not the total amount of cars or persons travelling over that link. For one not all persons will have a mobile phone, or will have their mobile phone turned on. And on the other hand there might be more than one mobile phone travelling in a car.

A possible solution for this problem is to adjust the number of routes. However adjusting the routes is a tricky operation. The datasets that were used for generating the routes do not have any more information. Therefore the additional information that is needed for an adjustment will have to come from another source. And mixing datasets from multiple sources is something that is often frowned upon.

But without any adjustment the routes will not hold much information, as they will never predict the total flow on a link (or, if they do, the user cannot be certain that it actually is the total flow). Therefore it is assumed that there will be users interested in an adjustment of the routes, and this section will describe how such an adjustment might work.

There are two ways to handle the adjustment of routes: with a factor based on survey data, or by using traffic or public transport counts as restrictions on the routes. These two approaches are described below in more detail. The approaches below will be explained in the situation of mobile phones in cars, but they are also easy to adapt to mobile phones in the other modes.

12.2.1 Factor based on surveys

The easiest adjustment is of course just to multiply all routes with a factor that indicates how many active mobile phones there are in a car. For this factor some information is needed that is easily obtained by surveys, and is therefore probably already available somewhere. The calculation of an adjustment factor is as follows:

$$Phones_{\text{on,vehicle}} = NofOccupants \cdot Pct_{\text{phones/person}} \cdot Pct_{\text{phones on}}$$

with $Phones_{\text{on,vehicle}}$ the number of active phones in a car, $NofOccupants$ the number of people in a car, $Pct_{\text{phones/person}}$ the percentage of people that have a mobile phone and $Pct_{\text{phones on}}$ the percentage of mobile phones actually turned on.

In addition to the above formula there are two more multiplication factors that have to be taken into account. If the original data is from only one mobile phone provider, the adjustment factor should also be multiplied with the market share of that provider, to compensate. And in the case of GPS mobile phones it should be multiplied with the percentage of phones that actually have a GPS locator installed (as not all mobile phones have a GPS locator).

Using the adjustment factor has several assumptions that might not be correct. For one it assumes that the numbers that are used are entirely correct, while such numbers are often dated. Secondly it also assumes that the dataset contains a perfectly represented distribution of mobile phones. And the distribution is exactly the same on all roads. These are of course assumptions that aren't true in the real world, and therefore the quality of the adjusted routes will be lower if an adjustment factor is used.

12.2.2 Counts as Restrictions

For some networks it might happen that there are traffic counts available. The availability of counts is quite likely as normal OD matrix calibration is often done with traffic counts. With systems as the NDW ([NDW, 2010]) traffic counts are becoming more and more available for studies and traffic modelling. Therefore a solution is to use these traffic counts to adjust the routes. OD matrix calibration already works in such a way, and the same approach can of course also be used to adjust the generated routes.

A route contains a list of all links that it traverses, and the number of phones that use that specific route. A traffic count belongs to a specific links, and has a number of cars passing by. Together a traffic count can therefore be used to adjust the routes passing the link, in a way similar as was described in section 3.6:

$$NC_k = NP_k \cdot \left(\frac{C_r}{\sum\limits_{j \in r} NP_j} \right)^{\epsilon_r}$$

with NC_k the real number of cars traversing route k, NP_k the number of phones traversing route k, C_r the total value of count r and ϵ_r the elasticity of count r. In the above formula the reference $j \in r$ means all routes r that pass the count r.

The above calculation has to be done for every route and count in the network. As described in [Smits, 2010] the normal way of implementing this, by taking the adjustments one after another, is not very good. Therefore all adjustments on one route will be taken together, and the geometric mean of the adjustment factors will be used:

$$NC_k = NP_k \cdot \left[\prod_{r \in R_k} \left(\frac{C_r}{\sum\limits_{j \in r} NP_j} \right)^{\epsilon_r} \right]^{1/|R_k|}$$

with NC_k the real number of cars traversing route k, NP_k the number of phones traversing route k, R_k the set of traffic counts attached to route k, C_r the total value of count r and ϵ_r the elasticity of count r.

With the routes adjusted it now is time to use the routes for the actual OD matrix calibration.

12.3 OD Matrix Calibration

At the end of the processing stage of the routes, these routes can finally be used for the actual purpose of this thesis: OD matrix calibration. Before the actual calibration is discussed, an overview is given of the steps taken until now:

- The original dataset has been processed by the user,
- For a GSM dataset areas have been defined,
- For a GPS dataset coordinates have been converted,
- The areas/coordinates have been converted to a set of routes between centroids,
- The routes have been split by mode,
- (possibly) The routes have been adjusted.

The routes found by the above steps consist of the following parts: an origin and destination (= two centroids), the path between the origin and destination (= a list of sequent links), a mode, and the number of mobile phones, cars or PT passengers traversing the given route. This last part is dependant on the fact if the routes are adjusted or not. If they are the number of cars or PT passengers is given, otherwise it is still the number of mobile phones. Therefore also the OD matrix calibration will be treated in two ways: either with adjusted routes, or without.

12.3.1 Calibration With Adjusted Routes

With adjusted routes, the routes give information about all displacements in that particular mode, from one centroid to another. Adding up all routes that start at the same centroid and end at the same centroid gives all displacements between two centroids. Because the routes are adjusted, the number of displacements is fairly certain to be exactly the total number of people travelling between the two centroids, and therefore the routes can be implemented directly as a general restriction:

$$g_{ij}^{new} = g_{ij}^{old} \cdot \left(\frac{C_r}{g_{ij}^{old}} \right)^{\epsilon_r}$$

The above definition is similar to the definition of an 1×1 block, as described in section 3.6.

As mentioned before it is often frowned upon to use another source of data to adjust a dataset. But still the use of such a combination of data sources can give good results. Below there are some requisites that will help with improving these results:

- The two datasets should come from a similar time period and day,
  Not that data from weekdays is adjusted with data from the weekend, or that data from one morning peak hour is adjusted with data averaged over a whole week.

- Every route should be adjusted at least twice,
  Not that every route is just dependant on one traffic count, that would probably give very bad results.
- Links with multiple routes should also be part of the adjustments,
  That way different routes will be adjusted together, meaning that the final routes will approximate the entire network, not every route will approximate one or two of the links it passes.

One note still has to be made. The adjusted routes give the amount of cars travelling between two centroids. But as mentioned earlier, not every car will hold one person. And as the OD matrices are often based on socio-economic data, which is represented by people, there is still another adjustment that has to be made. This adjustment is to convert the number of cars to the number of people. This conversion is something that is left to the devices of the user. With normal OD matrix calibration the fact that cars are not people is something that also has to be taken into account, so users will be familiar with this problem, and will have their own solution for it.

12.3.2 Calibration With Unadjusted Routes

When working with unadjusted routes the information given by the routes is incomplete. It is only known that there were a number of phones going from one centroid to another. But it is not known which fraction of the total traffic is modelled by these routes. Therefore the approach used in the previous section cannot be used for these routes.

The solution is to use the unadjusted routes for a “minimal calibration”, similar to the one described with Public Transport modelling, section 7.3.1. If the current value of an OD pair is higher than the value of the route, the route is skipped. But if the value of the route is higher than the value of the OD pair it is used as a normal calibration factor. In formula:

$$g_{ij}^{new} = \begin{cases} g_{ij}^{old} \cdot \left(\frac{C_r}{g_{ij}^{old}}\right)^{\epsilon_r} & \text{if } C_r > g_{ij}^{old} \\ g_{ij}^{old} & \textit{otherwise} \end{cases}$$

The above formulation gives rise to the following definition of the solution methods:

(for the adapted OmniTRANS algorithm:)

$$g_{ij}^{new} = g_{ij}^{old} \cdot \left[\prod_{r \in R^+} \left(\frac{C_r}{\sum\limits_{i,j \in r} g_{ij}^{old} \cdot P_{ijr}} \right)^{\epsilon_r \cdot P_{ijr}} \right]^{\frac{1}{\sum_{r \in R^+} P_{ijr}}}$$

with R^+ defined as $R^+ = \left\{ r \in R \;\middle|\; C_r > \sum\limits_{i,j \in r} g_{ij}^{old} P_{ijr} \right\}$.

(and for the gradient descend method:)

$$\min_{\mathbf{g}} F(\mathbf{g}) = \alpha \sum_{\substack{i,j \in I \\ d \in D}} \frac{1}{2} \left(g_{ij}^d - \hat{g}_{ij}^d \right)^2 + (1-\alpha) \sum_{r \in R^+} \frac{\epsilon_r}{2} \left(\sum_{\substack{i,j \in I, \\ d \in D_r}} P_{ijr}^d g_{ij}^d - C_r \right)^2$$
$$\text{s.t.} \quad \mathbf{g} \geqslant 0$$

with R^+ defined as $R^+ = \left\{ r \in R \,\middle|\, C_r > \sum_{\substack{i,j \in I, \\ d \in D_r}} P_{ijr}^d g_{ij}^d \right\}$.

12.3.3 Saving the Routes

The above two sections finalize the OD matrix calibration. But there is still one aspect which can be looked at. At the start of Chapter 10 it was noticed that "*where OV-Chipkaart data has information on the start and end of a trip, and not the route, GSM and GPS data has only the route, but often no clear start or end*". By attaching a beginning and an end to a route OD matrix calibrations became possible. But the route itself is also of importance. Perhaps not for OD matrix calibration, but for the general area of traffic modelling it is very important. That is why during the implementation a special look will be given into these routes, and the possibilities on how they can be used to improve the model.

With this final step the modelling of mobile phone datasets is complete. In the next chapter an overview will be given of how the GSM algorithm and the route processing from this chapter were implemented in a proof-of-concept prototype.

13. Implementation

This chapter describes the implementation of the mobile phone prototype. Only the basic implementation process will be discussed here, details will be skipped. First in section 13.1 the implementation of the GSM algorithm will be treated. And in section 13.2 something will be said about the attempted GPS implementation.

13.1 GSM Implementation

The first prototype that was started is the implementation of the GSM algorithm. For the GSM algorithm several steps had to be taken. First the measurements from the GSM dataset have to be converted to areas. This was easily done with a query on the coordinates, all nodes in OmniTRANS have their own coordinates. There were four possibilities for the shape of the area:

	Known aspects	Requirements
circle	range r_1	$d_Q \leqslant r_1$
ring	upper and lower range r_1, r_2	$r_2 \leqslant d_Q \leqslant r_1$
sector	range r_1 upper and lower angle ϕ_i, ϕ_2	$d_Q \leqslant r_1$ $\phi_2 \leqslant ang_Q \leqslant \phi_1$
sector part	upper and lower range r_1, r_2 upper and lower angle ϕ_i, ϕ_2	$r_2 \leqslant d_Q \leqslant r_1$ $\phi_2 \leqslant ang_Q \leqslant \phi_1$

with $d_Q = d(Q, BTS)$, the distance between the node Q and the base transceiver station BTS, and $ang_Q = \angle(Q, BTS)$, the angle of the node Q relative to the base transceiver station BTS.

The next step in the algorithm was to find the shortest paths between sequent areas. For this the shortest path engine in OmniTRANS was examined. At the first glance it worked very well, but later on it was discovered that the engine worked differently than expected. As mentioned in Chapter 11 the engine should find all shortest paths between the nodes in the total network. The paths would then be split by mode in a later stadium.

But the shortest path engine only works for the network of a specific mode. Therefore, if the shortest path between two nodes has to be found, first a mode has to be chosen. In one aspect this is very helpful, as it means that a route can be found per mode, and therefore the modal split becomes redundant. But on the other hand it also means that the number of times that the shortest path engine is used quadruples (with four modes), which means that the total runtime will grow very fast.

Another problem appeared soon after, also following from the different shortest path engine. The public transport network does not consist of just the transit lines in the network, but also every road that can in fact support public transport. Which means that a path through the public transport network does not have to follow a transit line, or even come close to one. Routes generated by the public transport network therefore are not usable, as they might indicate a public transport route, while it does not exist.

Despite these drawbacks the implementation was continued, and worked quite good. During some simple tests it was noticed that the memory needed would sometimes become very large. This was because every path is kept in memory during the algorithm, which takes a lot of space. So instead of keeping all paths and only filter them at the end, during the algorithm also a filter was implemented. When the different shortest paths are connected, to find the continuation of those paths, it could happen that there were multiple paths with the same starting node *and* end node. But from these paths only the best one is of interest, and therefore the other paths can just be deleted.

The fact that the above filter is allowed, is due to the fact that the 'cost' of the entire path is not dependant on how the first part of the path looked like. This was first defined by Bellman in his "Principle of Optimality" ([Bellman, 1957]). The principle says that "*an optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision*". An example:

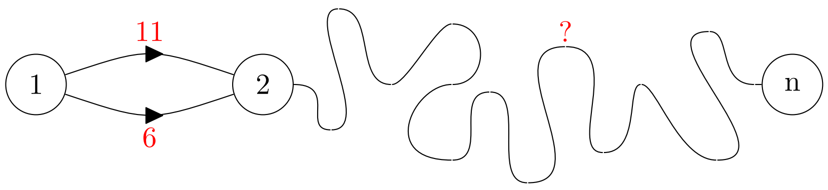


Figure 13.1: Two parts of an optimal route

In the above figure there are two paths between nodes 1 and 2. Now decisions in the path from node 2 to node n are not influenced by the choice how to get to node 2. The only important fact is that the path arrives at node 2. Therefore just the shortest path between nodes 1 and 2 can be taken, which is the lower path, with cost 6.

With the algorithm so far for every mode the shortest route is found, following the given GSM areas. But there is one aspect that is not yet taken into account: the time between the different measurements. These times indicate which mode was actually used. Therefore a check was added to the algorithm, to see if the time difference in the measurements match with the time it takes to travel the found path. The user can then define bounds on how much the time can deviate from the real time, and with that the modal split is immediately enforced, as walking, cycling and driving speeds differ enough to see which path belongs to which mode.

After the check on the times the GSM prototype was declared frozen. There were some improvements from section 11.1.4 that were not yet added, but first a look was taken into the GPS algorithm and modifications.

13.2 GPS Implementation

After the above GSM prototype was implemented the final date of this thesis was approaching rapidly. Therefore only a small look was given into the implementation of the GPS modifications. These modifications, as they are described in Chapter 11, were designed to ensure that the found routes from the GPS algorithm can also be used for OD matrix calibration. The first step therefore was to take a look at the current implementation of the GPS algorithm.

In enquiring after the GPS algorithm the problems began to appear. The algorithm was originally implemented in an older version of OmniTRANS, and used aspects of OmniTRANS that were changed. This meant that the algorithm itself would have to be changed as well. Also the algorithm was not immediately available, which further hindered the implementation. The final problem came when it became obvious that some parts of the GPS algorithm were hard-coded to belong to the older OmniTRANS version, meaning that the whole algorithm has to be rewritten if it is to be used for this thesis.

With these problems it became impossible to create a prototype using GPS coordinates in the time allotted, and the actual prototype was abandoned. But this does not mean that nothing can be proven. The modifications, if they were implemented, would have given routes similar to the routes from the GSM implementation. Therefore the inclination was already to recycle the calibrating part of the GSM prototype for the GPS implementation. Which means that by assuming that a GPS implementation exists, it can be used in exactly the same way as the GSM prototype. And therefore if the GSM prototype works, then also the GPS prototype would have worked.

With that in mind the next chapter will introduce some tests for checking the GSM prototype, and the results from those tests.

14. Results

In this chapter the results of the GSM prototype are treated. In section 14.1 two different test networks are defined, for the tests. In section 14.2 the different tests are defined, consisting of a proof-of-concept test and several limitation tests. Then in section 14.3 the results from these tests are given, and in section 14.4 some conclusions from the results are treated.

14.1 The Test Networks

For testing the GSM prototype a testing network is needed, to see if the used algorithm works, and how good it is. As with the testing of the public transport prototypes (Chapter 9) two networks are used, which are described below.

14.1.1 Simple Network

The first network is based on the "Simple model" from [Smits, 2010], as shown in the following figure:

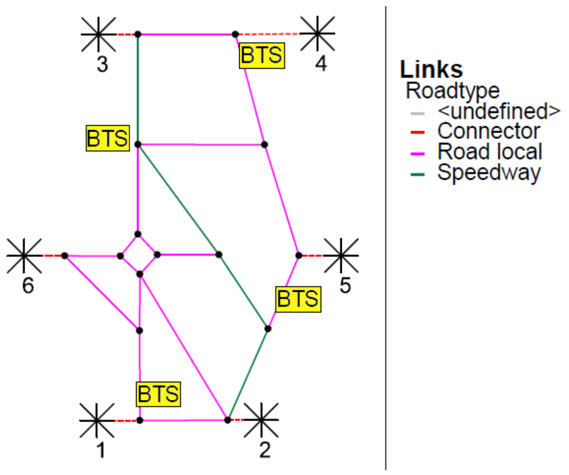


Figure 14.1: Simple Network

The simple network is build up of six centroids (the stars) and two road types, making it a small-case network. The highway differs from the other roads in that the cars using it can go twice as fast. In the network four GSM stations are placed, marked by the yellow BTS's (base transceiver station). The simple network will be used to test if the GSM prototype actually works, if it can generate routes and can use those routes for OD matrix calibration.

14.1.2 Delft Network

The Delft network is the standard tutorial network in OmniTRANS, consisting of large road network connecting together 22 centroids:

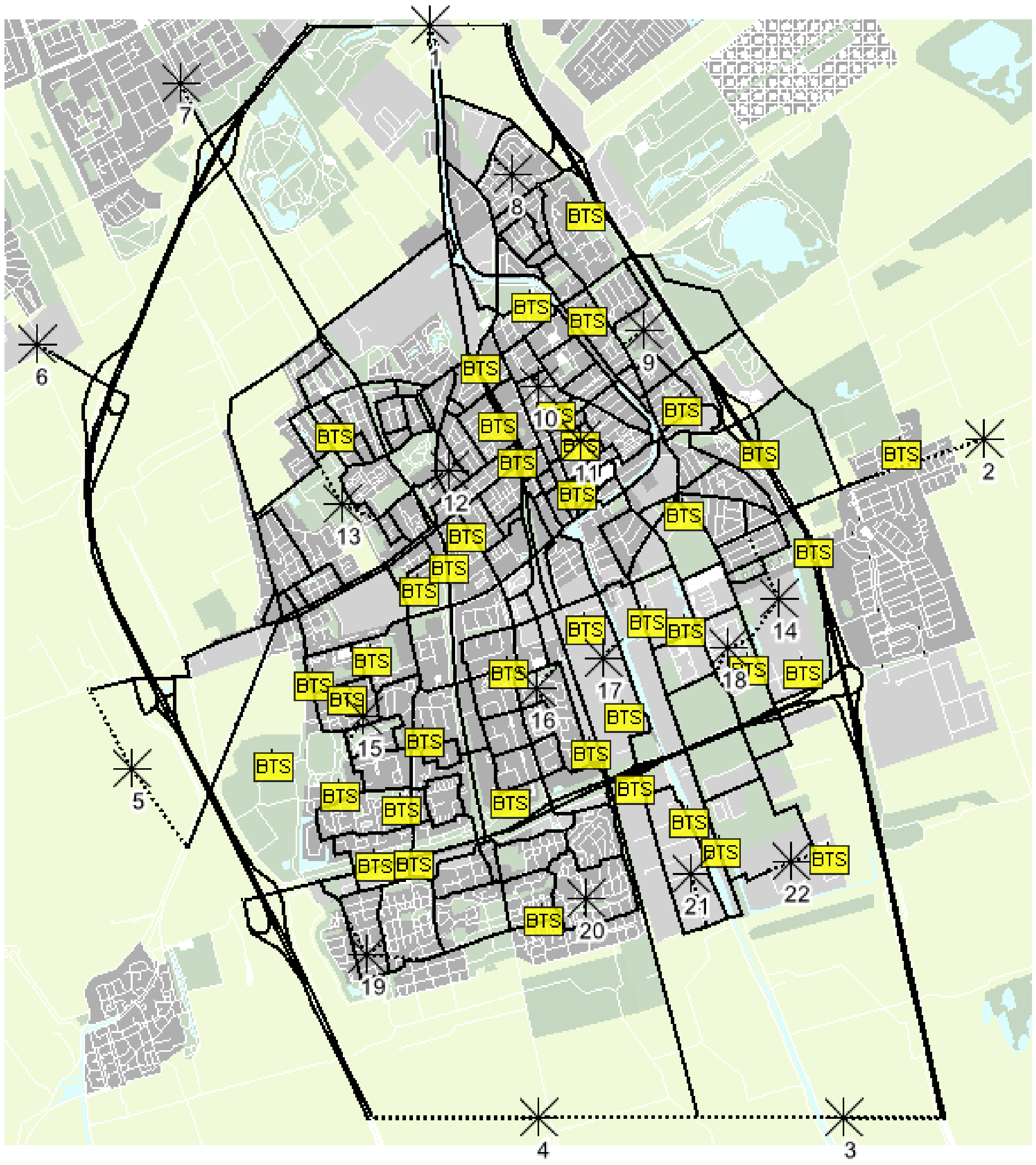


Figure 14.2: Delft Network

As can be seen from the above picture there are quite a few GSM masts in Delft (the yellow BTS's), which are not uniformly distributed over the whole network. The coordinates of these GSM transmitters were gotten from the "Antennebureau" ([Antennebureau, 2011]), a Dutch government agency that collects information and legislation about antennas in all categories (mobile phones, radio transmitters and even amateur transmitters). The transmitters used in the Delft network though are only the 'normal' GSM transmitters.

Sadly no real GSM data was available for this thesis. GSM datasets are quite privacy sensitive, and therefore cannot be obtained easily. For this thesis it was decided to again create some datasets by hand, for the different tests. More information about these datasets will be given in the next section.

14.2 Testing Options

With the two test networks given, the different tests can also be defined. For the public transport tests a lot of quality measurements were defined, to compare different results. But for the GSM prototypes the tests will test the prototype itself, and not the difference between multiple prototypes. Therefore the tests defined in this section will be of the type "succeed or fail", and no quality measurements are needed (or even possible).

In testing the GSM prototype there are two types of tests that have to be done. First a series of proof-of-concept tests have to be done, to check if the prototype actually works and calibrates the OD matrix successfully, based on the used GSM datasets. These tests will be done on the simple network, as then it is easy to check if the tests failed of succeeded. The second type of tests are limitation tests, tests designed to see what the limitations are of the prototype. These tests will be done on the Delft Network, trying to find all weaknesses and features of the GSM prototype, and through that of the modelling behind the prototype.

The tests for the simple network:

Test	**Explanation**
OD Calibration (PC1)	Supply two routes, and see if the OD matrix calibration is successful.
Route calibration (factor, PC2)	Supply two routes, calibrate them with a factor, and check if the OD matrix calibration is successful
Route calibration (counts, PC3)	Supply two routes, calibrate them with two counts, and check if the OD matrix calibration is successful
Modal Split (PC4)	Supply one route twice, with different speeds, and check the modal split.

Table 14.1: Tests for the Simple network

And the tests for the Delft network:

Test	Explanation
Standing Still (L1)	In a GSM dataset, have one BTS appear multiple times immediately after another.
U-turn (L2)	In a GSM dataset, switch between two BTS's, for example: going $A \rightarrow B \rightarrow A$.
Detour (L3)	In a GSM dataset, let the mobile phone make a detour, without this being registered.

Table 14.2: Tests for Delft Network

14.3 Results - Simple Network

The four proof-of-concept tests have been applied to the simple network. The used OD matrix contained a 1 for the OD pair (1,4), and 0 for all other OD pairs. The GSM dataset then contained two routes, one passing the upper left BTS, and one passing the lower right BTS, with both routes starting at centroid 1 and ending at centroid 4. Then the four proof of concept tests were run, to check the prototype. The used datasets can be found in Appendix D.

- **PC1 (OD Calibration)**
  The first test was completed successfully. The algorithm found the two routes without problem, and calibrated the OD matrix, changing the OD value to 2. For this calibration the Adapted OmniTRANS algorithm was used. When using the Gradient method for solving the OD matrix calibration the results were not so good, which in retrospect is quite logical. The Gradient method works by both using the restrictions as well as the old OD matrix. But in this case the old OD matrix is obviously wrong. Therefore the results of the Gradient method are best when $\alpha = 0$, then the OD value is 2. For other values of α the OD values are lower:

α-value	0.01	0.1	0.25	0.5	0.75	0.9	0.99
OD value	1.99	1.90	1.75	1.50	1.25	1.10	1.01

  This result means that for the other tests only the Adapted OmniTRANS algorithm is used.

- **PC2 (Route Calibration, factor)**
  In the second test the two routes from the GSM dataset are calibrated with a factor. This factor was set to 5, which meant that every route was multiplied with 5. And indeed, after the OD matrix calibration the OD value became 10, indicating that the second test was also successful.

- **PC3 (Route Calibration, counts)**
  The third test was similar to the second one, only now the two routes were calibrated with the counts in the network, and not a factor. The route going left had a count with value 1, the route going left had a count with value 9, together making 10 again. The OD matrix calibration went exactly like before, and again the OD value became 10, passing the test.

  (The situation where only one route was calibrated by a count was also looked at. In that case the other route would be multiplied with 0 in the calibration, resulting in it being deleted. This quite strongly shows the risk of using traffic counts to calibrate the generated routes, and shows the need for at least one count for every route!)

- **PC4 (Modal Split)**
  The fourth and final test in the simple network was to test the Modal split. To that end the times in the GSM dataset were greatly increased, to simulate walking times. Then the prototype was started as normal, with the OD calibration resulting in 2 walking people, as was expected. And with that the GSM prototype passed all four proof-of-concept tests.

In all four tests as described above the total runtime was in the order of 3 seconds, which is quite reasonable for such a small network. When the GSM dataset was copied five times, to create ten routes, the total runtime only increased to around 7 seconds, indicating that the route generation part of the algorithm probably runs in sub-linear time. To really prove this more tests will have to be done, more about that will be investigated in the next section.

14.4 Results - Delft Network

The Delft network is much larger than the simple network, and therefore perfect for some more complex tests. Because the network differs from the simple network, also different datasets are needed. For these datasets first a standard route is generated, from one centroid to another. Then for tests L1 and L2 variations on that datasets are generated, which simulate a specific situation. For test L3 new datasets are generated, as that test works differently than the other two. All these datasets can be be found in Appendix D.

- **L1 (Standing Still)**
  In the first test one additional entry is added to the route, indicating that the mobile phone stayed in one place for five minutes (probably for a conversation), after which the route is continued identically to the original route. The GSM prototype sadly couldn't cope with this situation. The areas before and after this pause couldn't be connected with a 'car' shortest path, which meant that a car route was disregarded. But the other measurements did not match the other modes, so in total no route was found. In other words, the test L1 failed.

- **L2 (U-turn)**
  In the second test the GSM dataset was adapted in such a way that the route made a U-turn, it went from one area (A) to another area (B), and then back again to area A. This was done in two ways. First the two GSM stations were very close together, as if the mobile phone just switched between the two stations, without a real U-turn taking place. In the second case there was a real U-turn, with the mobile phone heading the wrong way and then returning to the correct route.

  The first test, with a 'fake' U-turn, the prototype shifted the route a bit, as the pause was still taken into account for the route, resulting in more travelled distance. This is often not a problem, as the real goal of the prototype is to find OD pairs for the OD matrix calibration. Only if the generated routes are used for other purposes it might become a problem. The second test, with a real U-turn, gave exactly the same route as the original route, the detour generated by the U-turn was cut out of the route (as the shortest path was taken, so the detour was removed). Again this is not so bad, only in the case that specific routes are needed it will give some problems.

- **L3 (Detour)**
  In the third test more detours are checked, now to check how the prototype handles detours that aren't registered in the dataset. For this test two new datasets were generated, consisting of only the first and last record of a car journey. The first dataset shows a car driving through the city, while the highway was faster. And in the second dataset the situation is reversed, the car follows the highway while going through the city would have been faster.

  In both situations the prototype did not find the real route, which is logical, as it only looks at shortest routes. In the first test no route was found at all. But in the second route the prototype generated a bicycle route, which matched the longer time of the car journey, by using a direct route. This is of course an unwanted consequence, as these kind of situations are very hard to check.

The first two tests had run times lying in the neighbourhood of 360 seconds. This is quite a rise as compared with the tests from the simple network. However when the same tests were run with different routes, the run times would fluctuate widely, making a good comparison very difficult. Therefore no clear impression can be given on the increase of time on different projects. From the algorithm itself however there are some factors that will influence the runtime:

- The size of the network
- The number of measurements in one journey
- The concentration of nodes in the network

14.5 Conclusions

With the above results now something can be said about using Mobile Phone datasets for OD calibration. In the conclusions below GSM and GPS will be treated separately, as there is a lot of difference between GSM datasets and GPS datasets.

For GSM datasets it is shown that the quality of the data is quite low. Every measurement from a GSM dataset only gives information on an area where the person has been, which makes finding a precise route very difficult. The prototype, though itself quite simple, shows this quite clearly: it is very easy to change one time stamp or remove one measurement and totally confuse the algorithm. But for OD matrix calibrations the route is of less importance than the start/end of a route. Therefore the conclusion is that the routes from GSM can still be used for OD matrix calibration, but not for much else. One note here is that adjusting GSM routes with traffic counts is of course out of the order, as the routes are also not good enough to be calibrated in such a way. Minimal calibration and using an adjusting factor are still fine.

Because of the bad quality of the information for the routes, another difficulty is the modal split. It becomes very difficult to find the precise mode from a GSM route. The difference between a bus journey and a car journey was already difficult to see, but if the route is also inaccurate, the difference becomes almost impossible to see. And with a faulty modal split the OD matrix calibrations will also give bad results. The conclusion is therefore that the modal split is unsuitable for GSM datasets. It is better to just perform a 'complete calibration', calibrate all modes together with GSM data, than use a modal split with a low quality.

For GPS datasets the quality of the data is much higher, which is especially shown in [Fafieanie, 2009]. This means that the routes found from a GPS dataset will be far more useful than from a GSM dataset. For one it will be possible to execute a good modal split on the routes from a GPS dataset, meaning that the OD matrix calibration will be better. Also, as the routes have such good quality, they can be used for other purposes than just OD matrix calibration. More about that will be treated in Chapter 16.

The current downside to the GPS modelling is that at the moment nothing has been tested yet. There is a good algorithm implemented, but due to improvements in the OmniTRANS software it does not work at the moment. Also nothing is yet available for the use of GPS routes for the modal split or for an OD matrix calibration. Still parts of the implementation for GSM datasets can be reused for GPS routes. This gives at least an indication that the results are expected to be good, and therefore a real implementation and tests are very much in order.

Part III

Final Remarks and Conclusions

15. Mathematical Remarks

In this chapter two mathematical remarks will be treated, as referenced earlier in this thesis. In section 15.1 the factor α from the gradient descend method will be treated, especially on how to choose an optimal alpha. And in section 15.2 a closer look will be taken at the whole gradient descend method.

With the modelling and the implementations of both Public Transport (Part I) and Mobile Phones (Part II) finished, it becomes time to draw up the final conclusions. Before that there are some remarks that have to be treated. These remarks were already mentioned during this thesis, with the comment that they would be discussed at a later point. And as this thesis is nearing the end, it is time to discuss these remarks. In this chapter two mathematical remarks about the gradient descend method will be treated, three more general remarks will be treated in the next chapter.

15.1 Calculation of alpha

In using the gradient descend method first a factor α had to be given, as it was needed for the objective function. A reminder:

$$\begin{aligned} \min_{\mathbf{g}} F(\mathbf{g}) &= \alpha \sum_{\substack{i,j\in I\\ d\in D}} \frac{1}{2}\left(g_{ij}^d - \hat{g}_{ij}^d\right)^2 + (1-\alpha)\sum_{r\in R}\frac{\epsilon_r}{2}\left(\sum_{\substack{i,j\in I,\\ d\in D_r}} P_{ijr}^d g_{ij}^d - C_r\right)^2 \\ \text{s.t.}\quad & \mathbf{g} \geqslant 0 \end{aligned}$$

As noted in both section 9 and in [Smits, 2010] it is quite difficult to find a good α. For every new OD matrix calibration also a new α has to be determined, by running the calibration for different values of α and picking the best one. But in the modelling phase it is unknown what a good result is, and therefore it is also unknown which α will lead to that good result.

In the above objective function α determines how much the first part (the difference between the starting matrix and the current matrix) is weighted relative to the second part (how much the current matrix differs from the restrictions). From this there are three aspects that influence what would be a good choice for α:

- **Reliability of the starting matrix**
  The starting matrix is the matrix that is known before the calibration is started. This matrix can for example be the OD matrix of the year before, or generated by an algorithm like OtGravity (see section 2.1.2). But because the starting matrix is taken into account into the OD matrix calibration the reliability of the starting matrix is very important. If the starting matrix is known to be quite close to the 'real' OD matrix α should be high. But if the user has little confidence in the starting matrix the α should be low, as not to influence the calibrations too much.

- **Proportion of restrictions versus OD pairs**
  There are a number of OD pairs that "participate" in an OD matrix calibration, as well as a number of restrictions. Because both OD pairs and restrictions have a place in the above objective function, it is important to their relative sizes. If for example there are a lot of restrictions, and only a few OD pairs, the restrictions will dominate the objective function. In that case it can be better to raise the value of α, to compensate and restore balance. The same holds if there are only a few restrictions and a lot of OD pairs, then the value of α should be lower.

- **The total number of people**
  The third aspect has to do with the size of the calibration. If a small network is calibrated, with for example only 10 people that have to be assigned, the influence of *alpha* is bigger than with a network where 10 million people have to be assigned. Therefore the number of people in the OD matrix calibration give an indication of how detailed the α has to be.

The first two aspects can be used together to give a first indication of a 'good' α, a concept $\tilde{\alpha}$ so to speak. This can be done with the following approach:

- Rate the starting matrix with a number between 0 and 50, with 50 being "(almost) perfect".
- Find the proportion of restrictions relative to the number of (non-zero) OD-pairs, and multiply it with 50.
- Add the two numbers together, and divide by 100.

Or, in formula:

$$\tilde{\alpha} = \frac{\max\left\{B + \left(1 - \frac{|\mathbf{g}|}{|R|}\right) \cdot 50, 0\right\}}{100}$$

with $\tilde{\alpha}$, $B \in [0, 50]$ the rating of the starting matrix, $|\mathbf{g}|$ the number of (non-zero) OD pairs in the OD matrix, and $|R|$ the number of restrictions.

The above $\tilde{\alpha}$ will give the user a starting point where to start looking. Then the user can look in the neighbourhood of $\tilde{\alpha}$ to find the optimal α for their specific model.

15.2 Gradient Descend Method

For the OD matrix calibration in OmniTRANS [Smits, 2010] devised a gradient descend method approach for OmniTRANS. This method has led to a problem of the form

$$\min_{\mathbf{g}} F(\mathbf{g}) = \alpha \sum_{\substack{i,j\in I\\ d\in D}} \frac{1}{2}\left(g_{ij}^d - \hat{g}_{ij}^d\right)^2 + (1-\alpha)\sum_{r\in R}\frac{\epsilon_r}{2}\left(\sum_{\substack{i,j\in I,\\ d\in D_r}} P_{ijr}^d g_{ij}^d - C_r\right)^2 \qquad (*)$$
$$\text{s.t.}\quad \mathbf{g} \geqslant 0$$

15.2.1 General Description

The above problem is a quadratic optimization problem, of the general form

$$\min_{\mathbf{g}} F(\mathbf{g}) = \frac{1}{2}\mathbf{g}^T A\mathbf{g} + \mathbf{g}^T b + c$$
$$\text{s.t. } \mathbf{g} \geqslant 0$$

This can be easily seen in the following calculations:

$$\begin{aligned} F(\mathbf{g}) &= \alpha \sum_{\substack{i,j\in I\\ d\in D}} \frac{1}{2}\left(g_{ij}^d - \hat{g}_{ij}^d\right)^2 + (1-\alpha)\sum_{r\in R}\frac{\epsilon_r}{2}\left(\sum_{\substack{i,j\in I,\\ d\in D_r}} P_{ijr}^d g_{ij}^d - C_r\right)^2 \\ &= \frac{\alpha}{2}\|\mathbf{g}-\hat{\mathbf{g}}\|^2 + (1-\alpha)\sum_{r\in R}\frac{\epsilon_r}{2}(P_r^T\mathbf{g} - C_r)^2 \end{aligned}$$

Here $\mathbf{g}$, $\hat{\mathbf{g}}$ and P_r are all column vectors, with length $|I|\cdot|D|$. P_r contains the values P_{ijr}^d, when $d \notin D_r$ then $P_{ijr}^d = 0$. Now the first and second derivatives of $F(\mathbf{g})$ become:

$$\begin{aligned} \nabla F(\mathbf{g}) &= \alpha(\mathbf{g}-\hat{\mathbf{g}}) + (1-\alpha)\sum_{r\in R}\epsilon_r P_r(P_r^T\mathbf{g} - C_r) \\ &= \left(\alpha I + (1-\alpha)\sum_{r\in R}\epsilon_r P_r P_r^T\right)\mathbf{g} + \left(-\alpha\hat{\mathbf{g}} - (1-\alpha)\sum_{r\in R}\epsilon_r C_r P_r\right) \end{aligned}$$

$$\nabla^2 F(\mathbf{g}) = \alpha I + (1-\alpha)\sum_{r\in R}\epsilon_r P_r P_r^T$$

Therefore $F(\mathbf{g})$ can be written as: $F(\mathbf{g}) = \frac{1}{2}\mathbf{g}^T\left(\alpha I + (1-\alpha)\sum_{r\in R}\epsilon_r P_r P_r^T\right)\mathbf{g}$
$- \mathbf{g}^T\left(\alpha\hat{\mathbf{g}} + (1-\alpha)\sum_{r\in R}\epsilon_r C_r P_r\right) + \frac{\alpha}{2}\|\hat{\mathbf{g}}\|^2 + (1-\alpha)\sum_{r\in R}\frac{\epsilon_r}{2}C_r^2.$

15.2.2 Solving

From the above calculations it can be seen that $\nabla^2 F(\mathbf{g}) > 0$ (if $0 \leqslant \alpha \leqslant 1$, which is assumed to be the case), so it immediately follows that $F(\mathbf{g})$ is convex. The unconstrained quadratic problem, i.e. without the demand that $\mathbf{g} \geqslant 0$, can be solved quite easily, by just solving $A\mathbf{g} = -b$ for $\mathbf{g}$, and only checking afterwards if the resulting $\mathbf{g}$ is positive.

If $\mathbf{g}$ is not positive then the normal gradient descend method, as described earlier, can be used to find a non-negative $\mathbf{g}$. The steps in the gradient descend method take the following form:

step k: given $\mathbf{g}_k$, compute the minimal t_k of

$$\min_{t_k \geqslant 0} F(\mathbf{g}_k + t_k d_k), \text{ with } d_k = -\nabla F(\mathbf{g}_k)$$

Then take $\mathbf{g}_{k+1} = (\mathbf{g}_k + t_k d_k)^+$.

The $(.)^+$ was already mentioned in Chapter 4, as it is needed to keep the final solution non-negative. This approach will give a good result, due to the fact that the gradient is used as search direction. This implies that the solution $\bar{\mathbf{g}}$ of $(*)$ is a fixed point of the above algorithm. This can be seen as follows:

For $(*)$ the KKT-conditions (Karush-Kuhn-Tucker conditions) give the following (sufficient and necessary) conditions:

$$\bar{\mathbf{g}} \text{ solves } (*) \Leftrightarrow \nabla F(\bar{\mathbf{g}}) \geqslant 0 \text{ and } \bar{g}_i \frac{\partial F(\bar{\mathbf{g}})}{\partial \bar{g}_i} = 0$$

Then with these conditions the gradient direction d_k in the point $\bar{\mathbf{g}}$ becomes:

$$d_k = -\nabla F(\bar{\mathbf{g}}) \leqslant 0$$

and for $(\bar{\mathbf{g}})_i > 0$ this becomes:

$$(d_k)_i = -\frac{\partial F(\bar{\mathbf{g}})}{\partial \bar{g}_i} = 0$$

So for $\mathbf{g}_{k+1} = (\bar{\mathbf{g}} + t_k d_k)^+$ we have:

$$(\mathbf{g}_{k+1})_i = \begin{cases} 0 & \text{if } (\bar{\mathbf{g}})_i = 0 \\ \bar{\mathbf{g}}_i & \text{if } (\bar{\mathbf{g}})_i > 0 \end{cases}$$

And therefore $\bar{\mathbf{g}}$ is a fixed point of this iteration of (*). This can also be seen in the following figure:

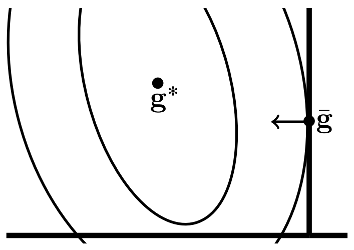


Figure 15.1: Fixed point $\bar{\mathbf{g}}$

The search direction from $\bar{x}$ will always be perpendicular to the vertical axis, and therefore the truncation will return to the point $\bar{x}$ again.

In section 3.7 it was noted that [Smits, 2010] changed the gradient descend method, to make sure that the OD pairs with a 0 also remained 0. This was done by multiplying the search direction d_k with the current OD matrix $\mathbf{g}$. But this approach has a very large downside to it. When during the solving of the gradient descend method an OD pair becomes 0, this will mean that that OD pair will remain 0 for the rest of the calculations. This can be seen in the following figure:

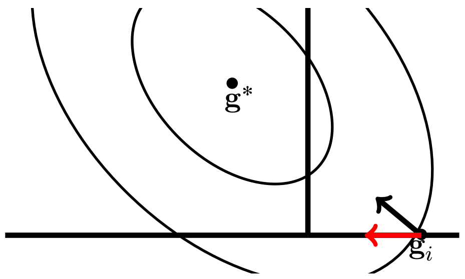


Figure 15.2: Downside

In the above figure the point $\mathbf{g}_i$ is the current solution. For the next iteration, $i+1$, the search direction is given in black. But as the point $\mathbf{g}_i$ is located on the horizontal axis, it means that only the horizontal direction will be taken into account, as the vertical direction will be multiplied with 0. This new search direction is given in red. In the above example the final solution will probably end in the point (0,0), as then the search direction will be multiplied with 0, and it becomes a fixed point.

16. Other Remarks

In this chapter three general remarks will be treated. In section 16.1 the overlap between the public transport and the mobile phone models will be treated. Then in section 16.2 other uses for the routes from the GPS modelling will be treated. And in section 16.3 a small discussion of OD calibration versus OD estimation will be given.

16.1 Overlap

Up until now the areas of public transport and mobile phones have been treated separately. But of course there is some overlap, as there will be mobile phones travelling in the public transport also, as was described in section 12.1.

If the two models are compared, it immediately shows that the data from the OV-Chipkaart is more complete. A dataset from the OV-Chipkaart describes all public transport journeys in the network (of a certain transit provider), while a GPS or GSM dataset will always only show a fraction of these journeys, as not everyone has a mobile phone, and often a mobile phone dataset comes from only one provider. But on the other hand mobile phone datasets show a complete journey, while the OV-Chipkaart data only shows part of that journey.

This means that the two different models will have some information to add to each other, improving both models. First a look will be given into how the public transport can be improved by a mobile phone dataset, then the other way around.

16.1.1 Improving the Public Transport Model

As said before, the public transport model works with (incomplete) public transport journeys. In section 5.3.4 already a look was taken into connecting two of these journeys, two Tracks. The conclusion then was that it was impossible, as that information is not available. But by using a GPS dataset it might become possible to find this information after all.

Because a GPS dataset shows the entire journey of a mobile phone, from all these journeys it can be gathered how many people use multiple Tracks, and more importantly, which Tracks in particular. Now a GPS dataset does not give the complete picture, as not all people have a mobile phone. But it can give a good indication of which Tracks can be (partially) taken together, to improve the restrictions that flow from the Tracks.

Another aspect from a public transport that is not included in the OV-Chipkaart data is the mode of the access and egress flows. From the OV-Chipkaart data only the access stop and egress stop are known, not how the passenger got to that stop, or how he/she continued the journey. With a GPS dataset again an indication can be gotten of how these journeys look like and which mode was used.

16.1.2 Improving the Mobile Phone Model

Because the data from the OV-Chipkaart is so complete, it should always be used if it is available. So if a GPS dataset is also available it should mainly be used for calibrating the other OD matrices, for cars, bicycle and walking.

But the OV-Chipkaart data can still play some part in the mobile phone model. As mention in the problem description, the mobile phone routes are very important, as they might be used for other purposes than just OD matrix calibration. But to get good routes the routes from a GPS dataset have to be calibrated, as described in section 5.3.4. And these calibrations need the information from the OV-Chipkaart dataset, to generate GPS routes of high quality.

Using the GPS routes for other purposes is something that has not been studied yet in this thesis, and therefore this will be done in the next section.

16.2 GPS Routes

The routes that are generated from a GPS dataset show the entire journey of a mobile phone, from the origin to the destination, with any possible stops along the way. For OD matrix calibration the route is not very important, it is only useful in calibrating the routes, to convert the number of mobile phones following a route to the number of cars or people following that route. But for other modelling areas the GPS routes are very interesting. In this section three possible uses for GPS routes will be described, but many more are possible.

- **Route Set**
  First off GPS routes can be used for defining a route set. This is a set of routes that cars use in the network to get from one zone to another. A route set is used by the allocation algorithm, that allocates the OD flows to the road network. Normally allocation algorithms work with shortest paths, but as the route set is known to be used, it will give a much better result.

The usage of GPS routes for a route set is of course already treated in [Fafieanie, 2009], as that is were the algorithm for GPS datasets came from. Therefore the usage of GPS routes for a route is already proven and tested.

- **Checking New Situations**
  Secondly the routes from different GPS datasets can show the effect of certain changes in time and or infrastructure. For example two GPS datasets, one from before a major roadworks, and one after, can show if the road works really did improve the infrastructure and such. Or several GPS datasets can be used to check which alternate routes people take in the situation of extensive roadworks, to check if they use the recommended alternatives and such.

  Using GPS routes for checking how certain situations changed the traffic flows is of course a measure that can only be used afterwards, and is not very useful during the situation itself (or the process of delivering the data and processing it has to be done very fast), but more for afterwards. The conclusions from such can therefore better be used as recommendations for new projects and modellings.

- **Number of People in a Car**
  Finally there is a third aspect of OD matrix calibrations that can be improved by using information from the GPS routes. As mentioned in section 12.3 the OD matrix shows a number of people travelling between two centroids, and the routes are actually cars. Now, with a full GPS dataset (so from multiple mobile providers) the routes can be compared with traffic counts from the same time period to find a good approximation of the number of people in a car.

  At this moment the factor of how many people there are in a car is taken from some large surveys, which has a low quality compared with the above calculations. So by using GPS datasets and traffic counts a much better approximation can be found, which is also uwsefull for other parts of traffic modelling.

16.3 Calibration versus Estimation

In the course of this thesis there were times that the discussion about OD matrix calibration versus OD matrix estimation flared up, especially when talking with traffic modellers. In this thesis the focus has always lain with calibration, using restrictions to help improve an a priori matrix to get as close as possible to reality. But with the end of this thesis nearing, also a look will be given into OD matrix estimation.

OD matrix estimation is the process of designing the OD matrix from scratch, out of the information that is available. For good results to come from OD matrix estimation, the starting information has to be of a high quality. Now the datasets treated in this thesis are of that high quality. The GPS datasets give routes so precise that the entire route, with all possible stops, is generated. And data from the OV-Chipkaart gives all public transport journeys, for a specific transit provider, in the entire network.

Aside from the high quality there is another reason to look at OD matrix estimation. This comes from the fact that the restrictions formulated both by the OV-Chipkaart data and GPS data is so plentiful that the restrictions swamp the starting matrix. This problem was already discussed in part in section 15.1, with the calculation of a good α.

But with the amount of restrictions that can be generated by de given datasets, it can happen that α approaches 1. With so many restrictions the calculation of alpha has to be precise, a small deviation from the 'good' α would either result in not using the restrictions, or only using the starting matrix. And as mentioned before it is very difficult to find the correct α. Therefore this is a problem that definitely will appear in further studies, and will have to be taken into account.

Of course it is not a bad thing that the above happens. Because matrix estimation is needed, due to the size and quality of the restrictions, it also means that matrix estimation will give better results than matrix calibration. The only thing is that it needs a different approach in modelling and thinking. But that is something that most traffic modellers are getting used to anyway, in this age of information.

17. Conclusions and Recommendations

In this final chapter the final conclusions of this thesis will be treated, together with some recommendation based on these conclusions. In section 17.1 a small summary of the main results will be given, together with the conclusions drawn from these results. And in section 17.2 the recommendation based on these conclusions will be given.

17.1 Conclusions

The main research question of this thesis was described in chapter 1, and can be summarized in the following question:

"*How can OD matrix calibration be expanded with information from the public transport and from mobile phones?*"

In this section the results to this question will be treated. As both the public transport and mobile phone models were treated separately in the course of this thesis, also the results from these models will be treated separately.

17.1.1 Public Transport

Information from the public transport took the form of the OV-Chipkaart, which gives information on journeys made in the public transport network. The data from the OV-Chipkaart takes the form of a stop-to-stop matrix, showing the number of people travelling from one stop to another in a certain time period. The data from the OV-Chipkaart gave rise to three options. First the data can be converted so counts on the links in the network. The second option is to check all OD pairs using a stop, and calibrating those OD pairs. And thirdly the entire journey can be taken into account, and all OD pairs using that journey are calibrated.

All three options were implemented in Ruby, and run in OmniTRANS. Here it was noticed that the second and third option needed information about the paths used by different OD pairs, which was not available in OmniTRANS. Therefore the implementations of these two options did not work as good as planned. In testing the three implementations the first and third options gave very good results. Only the second option, calibrating the stops, gave very bad results.

The data from the OV-Chipkaart can therefore be used in two different ways. First the data can be converted to counts, which can be used in the OD matrix calibration in the same way as traffic counts are used at this particularly moment. The implementation of this option is simple, and the resulting counts will give much more information than was available until now.

The other option, using the entire journey from the OV-Chipkaart data to calibrate OD flows, still needs some implementation to work as planned. This comes from the fact that not all information needed for this implementation is available in OmniTRANS at the moment. But if this additional information is available the first option will give much better results than only the counts from the first option.

The two options can at the moment be used in two different ways. This comes from the fact that the OV-Chipkaart is not yet the only valid payment option in the Netherlands. First the two options can be used in a minimal calibration, only using the restrictions if they increase the OD value. And secondly the OV-Chipkaart data can be completed, with a factor based on the penetration of the OV-Chipkaart. When the OV-Chipkaart is rolled out completely the data can be used for a full OD matrix calibration.

17.1.2 Mobile Phones

The information from mobile phones comes from two sources: either GSM datasets, or GPS datasets. Both datasets give information on the location of a mobile phone, at certain time intervals. For GSM datasets these intervals are quite large, and the location is inaccurate, for GPS datasets the time intervals are small and the locations very accurate.

As one measurement itself cannot be used for OD matrix calibration the entire route of that mobile phone has to be found. For finding the route from GPS datasets already an algorithm exists, from [Fafieanie, 2009]. For the GSM datasets a simple algorithm was designed in this thesis, and implemented in Ruby. With the found routes also some adjustments had to be made. First the used mode of the routes had to be determined. Secondly the routes did not represent the total number of people travelling in a network, as not everyone 'real' person will have a mobile phone. Therefore the routes either have to be used for a minimal calibration, as described in section 12.3, or made complete. This can be done by multiplying the routes with a factor, or adjusting them with traffic counts from another source.

For the GSM datasets it was concluded that the data was of such poor quality that a modal split would be out of the question, and therefore only a complete calibration could be done (with all modes calibrated in one go). For the GPS algorithm that problem does not exists, but as the GPS algorithm was not operational no real study was made of that. The GSM algorithm performed not so well in the tests in OmniTRANS, as the algorithm was quite easily fooled with imperfect datasets.

Therefore it was concluded that the GSM datasets can be used for OD matrix calibration, but will not give very good results. The datasets from GPS datasets will give much better results, and are therefore preferred over GSM datasets.

17.2 Recommendations

With the above conclusions there is of course some room for improvement and side notes. Therefore in this section some recommendations will be given.

In comparing the public transport and mobile phone models, the public transport models were the most interesting and therefore are recommended to be implemented first. At the moment traffic modellers do not use the public transport models much, as it is quite difficult to get good OD matrices for them. With the information from the OV-Chipkaart that could quite easily change, and it will improve the traffic models greatly. The first options, with the counts, can be implemented quite easily, as no big adjustments are needed. For de second option, the Tracks, it is needed that the paths between different centroids are known, which is not available at this moment. But if that information is available it is not difficult to also implement the second option, which will also improve the OD matrix calibrations greatly.

For the mobile phone models it is quite important that the GPS algorithm is fixed so that it actually works again. At the moment there is no modal split available for the GPS algorithm, this is something that would have to be researched and implemented when the GPS algorithm is working again. The GSM algorithm and model should only be improved if there is a client willing to work with them. The quality of a GSM dataset will never come close to a GPS dataset, and therefore the preference should always lie with GPS datasets.

With the above recommendations there are of course still some areas and problems that have to be researched further. Therefore below a few aspects are treated that would give interesting follow-up studies.

17.2.1 Public Transport

- **Counts: Multiple Transit Lines**
  The problem that appeared at the implementation of creating counts in section 8.1 was that overlapping transit lines would confuse the algorithm. Now this is not only a problem of the implementation, as it is a problem in general. If there are two transit lines that pass by the same two stops from the matrix H there is no way to check which transit line was used. In the current implementation only the transit line that takes the shortest path between the stops is used. A better option might be to look at travel times to select a transit line, or select several transit lines with (almost) the same travel time and divide the passengers among these transit lines.

- **Tracks: Using a Choice Model**
  The problem with Tracks, as described in section 5.3.4 was that they only showed a part of a journey, and it would be better to connect different tracks together. In Chapter 16 one possible way of doing that, by using the routes from a GPS dataset. But that will not always be possible. Therefore it will be interesting to look at other alternatives. The OtTransit algorithm in OmniTRANS uses a choice model to determine the paths of passengers. So it might also be possible to connect different Tracks with a choice model, with some probability two Tracks are connected, or some percentage of both flows can be taken together. It might be interesting to see what the possibilities with that might be.

17.2.2 Mobile Phone

- **Breaking Routes**
  With the routes generated from mobile phone datasets there is an aspect that was not treated in this thesis, but is of some importance. This aspect is the breaking of routes, when a route actually consists of multiple smaller routes. If a mobile phone user travels from a village to the city to do some shopping, and then travels back home, this should be modelled as two distinct routes. And with mobile phones which are turned on all day it follows that a lot of routes are contained in one dataset. Therefore it is important to research how such breaks are best identified and modelled. Not all breaks will be as easily recognized as the one described above, and therefore a good approach will be crucial for generating good routes.

- **Modal Split**
  For the GPS algorithm, as implemented by [Fafieanie, 2009], still the modal split will have to be modelled and implemented. In section 12.1 some pointers were given on how the mode could be identified, but this will not be enough. There will always be routes where it is very difficult (or even impossible) to find the used mode. Therefore it might be interesting to do some research in this area. Foremost of course to find an approach that will identify the mode of almost all routes. But after that, it should also be looked at what should be done with the routes from which the mode could not be identified. It could be that those routes will have to be discarded, but perhaps also still some information can be gathered from those routes.

- **Additional Uses for Routes**
  As already described in section 16.2, the routes from mobile phone datasets can be used for other uses as well. The three options that were given are of course interesting to research further, but it might also be important to look at other uses, perhaps based on a survey for traffic modellers or an extensive literature research.

17.2.3 Other

- **Better Calculation of alpha**
  For the gradient descend method it might be interesting to do some more research on the exact calculations to find a good alpha. This is mainly an interesting aspect for the users of OmniTRANS, but it might be good to have some insight into this matter, to better be able to advise users on which values of alpha will give a good result for their model.

- **Keeping OD pairs 0**
  In sections 3.7 and 15.2 it was noted that in an OD matrix calibration it is preferred to have OD pairs that are 0 also 0 in the final solution. The approach of [Smits, 2010], multiplying the search direction with the current OD matrix does not work, and therefore another solution has to be found. The problem is of course that by taking some OD pairs 0 the search direction of the gradient descend method might be changed, which might have other implications. It should therefore be researched in some depth which would be the best solution to this problem.

A. Bibliography

Abrahamsson, T. 1998. *Estimation of Origin-Destination Matrices Using Traffic Counts: A Literature Survey.* Tech. rept. International Institute for Applied Systems Analysis. Working Papers.

Antennebureau. 2011. *Antenneregister.* (website, in Dutch). `http://www.antennebureau.nl/antenneregister`, last visit: 17-03-2011.

Bellman, R.E. 1957. *Dynamic Programming.* Dover Books on Mathematics. Dover Publications. Republished in 2003.

Chan, Joanne. 2007 (June). *Rail Transit OD Matrix Estimation and Journey Time Reliability Metrics Using Automated Fare Data.* M.Phil. thesis, Massachusetts Institute of Technology.

Cui, Alex. 2006 (September). *Bus Passenger Origin-Destination Matrix Estimation Using Automated Data Collection Systems.* M.Phil. thesis, Massachusetts Institute of Technology.

Dutch, Steven. 2010. *Converting UTM to Latitude and Longitude (Or Vice Versa).* (website). `http://www.uwgb.edu/dutchs/usefuldata/utmformulas.htm`, last visit: 01-12-2010.

Fafieanie, Mike. 2009 (September). *Calibrating route set generation by map matching GPS data.* M.Phil. thesis, Universiteit Twente. Project done at Goudappel Coffeng, Deventer.

GSM World - Market Data. 2010. *Market Data Summary.* (website). `http://www.gsmworld.com/newsroom/market-data/market_data_summary.htm`, last visit: 05-10-2010.

Minister C. Eurlings. 2010 (September). *Commissievragen OV-chipkaart.* Letter sent to the Chairman of the House of Representatives (in Dutch).

Ministry of Transport, Public Works and Water Management. 2008 (June). *Wat U Moet Weten Over De OV-Chipkaart.* (brochure, in Dutch).

NDW. 2010. *Nationale Databank Wegverkeersgegevens.* (website). `http://www.ndw.nu/`, last visit: 10-12-2010.

Omnitrans. 2010 (May). *OmniTRANS for Modellers.* A Three Day Course (compatible with version 5.1.14 or higher).

OmniTRANS. 2010 (July). *OmniTRANS Internal Manual.* (compatible with version 5.1.20).

Omnitrans International. 2010. *OmniTRANS - Transport Planning Software.* (website). `http://www.omnitrans.nl/`, last visit: 07-10-2010.

Ortúzar, Juan de Dios, & Willumsen, Luis G. 2001. *Modelling Transport.* Third edn. John Wiley & Sons, LTD.

OV-Chipkaart.nl. 2010. *Where can you travel with the OV-chipkaart?* (website). `http://www.ov-chipkaart.nl/allesoverdeov-chipkaart/waarreizen/waartegebruiken/?taal=en`, last visit: 04-10-2010.

Pelletier, Marie-Pier, Trépanier, Martin, & Morency, Catherine. 2009. Smart Card Data in Public Transit Planning: A Review. *CIRRELT*, **46**.

Quddus, Mohammed A., Ochieng, Washington Y., & Noland, Robert B. 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, **15**(5), 312 – 328.

Schilpzand, Maarten. 2009. *Technical Document Matrix Estimations.* Internal document OmniTrans, Deventer.

Smits, Erik-Sander. 2010 (June). *Origin-Destination Matrix Estimation in OmniTRANS.* M.Phil. thesis, Utrecht University. Project done at Omnitrans, Deventer.

Spiess, Heinz. 1990. *A Gradient Approach for the O-D Matrix Adjustment Problem.* Tech. rept. Centre de Recherche sur les Transports de Montréal. Publication 693.

van der Zijpp, N.J. 2005. *Floating car data voor DVM toepassingen.* Released by ModelIt (in Dutch).

Wikipedia. 2010. *Public Transport.* (website). `http://en.wikipedia.org/wiki/Public_transport`, last visit: 01-10-2010.

B. List of Figures and Tables

List of Figures

List of Tables

C. Test Networks Data - PT

C.1 Simple Network

The matrix H for the Simple network:
(a stop-to-stop matrix, therefore with 8 entries)

	1	2	3	4	5	6	7	8
1	0	150	50	75	50	0	0	100
2	150	0	475	25	75	0	0	50
3	50	475	0	75	100	0	50	0
4	75	25	75	0	75	0	100	0
5	50	75	100	75	0	0	0	100
6	0	0	0	0	0	0	250	150
7	100	50	0	0	100	150	0	0
8	0	0	50	100	0	250	0	0

Table C.1: Matrix H for the Simple Network

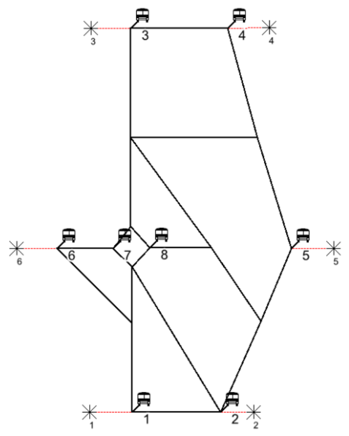


Figure C.1: Stop numbering

From the OD matrix calibrations several OD matrices were generated. Below is the best one, from using the Tracks prototype with the Gradient algorithm ($\alpha = 0,75$):

	1	2	3	4	5	6
1	0	100	100	75	50	100
2	100	0	400	50	75	50
3	100	400	0	50	100	50
4	75	50	50	0	75	100
5	50	75	100	75	0	100
6	100	50	50	100	100	0

(a) Real OD Matrix

	1	2	3	4	5	6
1	0	110,42	68,32	80,45	60,50	104,44
2	126,93	0	466,17	9,40	64,47	35,66
3	12,84	461,60	0	107,92	85,16	70,59
4	87,20	14,12	58,46	0	95,26	84,95
5	64,67	64,41	76,94	89,22	0	95,68
6	130,70	53,33	64,06	55,83	86,96	0

(b) Best approximated OD Matrix

Table C.2: Real and Approximated OD matrices for the Simple Network

C.2 Delft Network

On the next pages the used OD matrix and the best approximated OD matrix are shown (Tracks, $\alpha = 0,15$), for the Delft network. The used matrix H is not shown, as it is too large for this report.

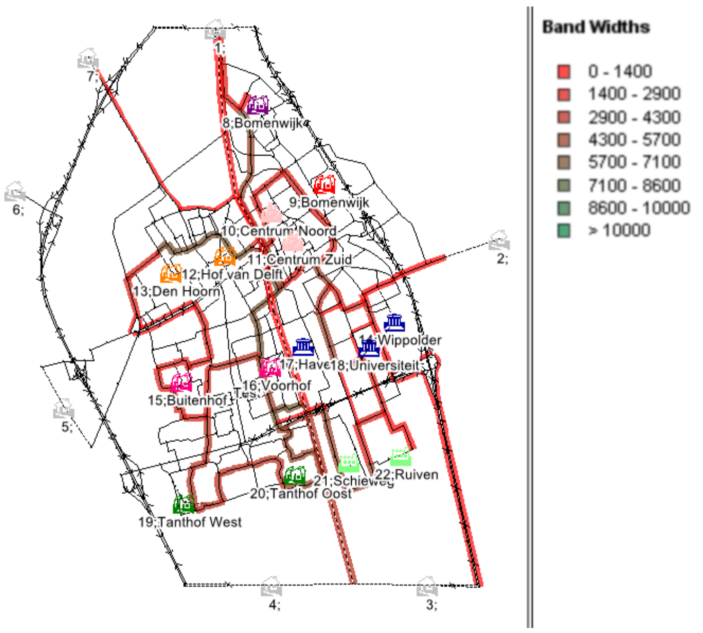
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0	0	0	0	0	0	587	0	200	50	0	0	0	0	0	0	100	0	0	0	0
2	0	0	0	0	0	0	0	0	350	0	0	0	0	249	0	0	0	101	0	0	0	0
3	0	0	0	0	0	0	0	0	0	101	92	0	0	0	0	0	0	0	0	104	0	0
4	0	0	0	0	0	0	0	0	0	99	108	0	0	0	0	0	0	0	0	96	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	83	10	48	47	0	0	42	0	52	300	122	0	208	28	243	250	0	201	0	0	50	56
9	100	50	44	50	0	0	56	259	0	391	290	514	200	51	100	205	50	249	100	100	150	141
10	147	0	97	103	0	0	42	450	50	0	253	300	292	126	31	100	0	300	0	0	140	194
11	144	0	106	100	0	0	194	382	600	98	0	518	100	0	27	16	0	238	100	300	100	112
12	439	10	302	311	0	0	350	391	548	311	278	0	300	122	23	580	0	0	0	111	539	150
13	54	0	25	75	0	0	250	47	603	133	399	751	0	148	25	500	0	0	0	67	448	100
14	0	43	0	0	0	0	0	296	504	67	350	495	48	0	0	505	19	112	0	233	0	88
15	511	120	785	289	0	0	150	304	596	306	250	568	0	40	0	950	100	200	114	142	560	400
16	139	120	203	200	0	0	136	318	584	95	93	482	100	84	179	0	95	222	0	200	540	300
17	0	0	0	0	0	0	0	0	0	50	60	50	0	0	50	90	0	195	0	0	0	0
18	0	0	0	0	0	0	0	0	0	400	401	999	70	0	0	0	0	0	0	0	0	0
19	0	0	48	242	0	0	0	10	0	345	255	200	350	100	501	350	650	199	0	489	411	350
20	0	0	142	158	0	0	0	223	97	350	245	391	0	0	0	0	505	250	250	0	290	299
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table C.3: Real OD Matrix for the Delft Network

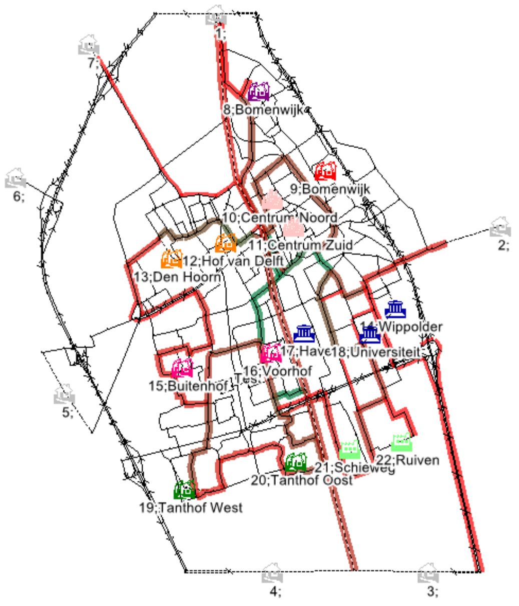
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0,02	105,58	73,01	0,00	0,00	2,52	735,34	0,48	0,00	0,00	3,12	1,15	0,00	2,32	12,61	0,50	11,75	0,81	12,51	0,18	0,00
2	12,95	0	67,44	42,95	0,00	0,00	15,22	57,30	1,48	0,00	0,00	5,90	2,92	0,00	6,40	24,31	4,63	469,61	0,61	9,01	1,08	0,03
3	58,34	0,07	0	20,38	0,00	0,00	5,62	24,42	0,28	0,00	0,00	2,54	0,57	0,00	4,39	41,88	0,92	113,46	0,82	43,83	0,69	0,01
4	76,12	0,10	46,76	0	0,00	0,00	7,97	34,03	0,31	0,00	0,00	3,48	0,47	0,00	6,04	55,18	1,24	32,43	1,13	58,96	0,96	0,02
5	0,00	0,00	0,00	0,00	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
6	0,00	0,00	0,00	0,00	0,00	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
7	0,00	0,00	0,00	0,00	0,00	0,00	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
8	1014,66	0,16	66,37	40,81	0,00	0,00	306,31	0	6,59	0,01	0,00	39,65	13,76	0,00	24,19	84,17	6,23	96,70	8,93	51,77	0,51	0,01
9	37,84	24,94	42,09	19,34	0,00	0,00	76,67	317,42	0	176,59	0,00	1116,62	20,06	630,13	4,07	16,76	481,45	129,57	1,47	29,31	0,98	0,04
10	42,21	15,25	25,69	27,08	0,00	0,00	167,42	347,58	0,80	0	311,89	1029,71	47,42	380,52	4,77	22,26	110,41	83,87	1,87	34,59	0,59	0,01
11	0,52	1,01	1,80	0,29	0,00	0,00	2,71	4,83	963,63	124,89	0	626,18	900,01	0,00	7,38	0,79	500,87	10,08	15,63	85,69	0,04	0,00
12	1,31	0,55	0,95	0,67	0,00	0,00	14,63	13,89	2208,31	1055,85	1131,35	0	260,53	0,00	3,46	43,44	0,08	2,63	0,71	97,11	0,02	0,00
13	16,27	9,50	42,44	25,20	0,00	0,00	130,31	134,21	38,94	1589,81	94,79	705,60	0	0,00	23,07	69,87	223,56	468,44	4,43	75,05	0,54	0,01
14	2,03	39,32	56,75	367,44	0,00	0,00	10,54	16,94	547,97	0,69	0,00	616,45	0,55	0	40,33	3,52	0,58	250,57	126,88	297,47	1,11	460,18
15	228,45	146,80	661,37	519,99	0,00	0,00	347,21	1099,30	26,96	3,88	389,38	346,06	234,73	0,00	0	936,42	32,17	158,76	106,83	155,73	982,98	32,42
16	63,35	27,89	379,67	280,49	0,00	0,00	80,92	268,75	1,12	0,00	0,00	378,81	13,63	0,00	294,92	0	4,79	61,52	21,58	345,50	1787,83	105,45
17	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	207,95	494,65	0,00	0,00	0,00	0,00	0,00	0,00	0	3,94	0,00	0,00	0,00	0,00
18	4,20	53,63	93,82	11,25	0,00	0,00	17,48	27,87	0,74	0,00	0,00	2,62	0,97	0,00	2,19	21,26	4,31	0	0,61	21,16	9,35	1632,00
19	11,55	0,02	20,12	14,45	0,00	0,00	2,35	90,71	0,26	0,00	1724,79	18,34	143,62	282,26	649,52	195,16	43,43	372,39	0	500,28	464,17	12,36
20	24,53	0,07	162,01	117,92	0,00	0,00	6,80	68,45	0,34	0,00	0,00	393,64	11,94	0,00	90,90	2001,78	3,50	79,04	258,25	0	10,62	0,20
21	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0	0,00
22	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0

Table C.4: Best Approximated OD Matrix for the Delft Network

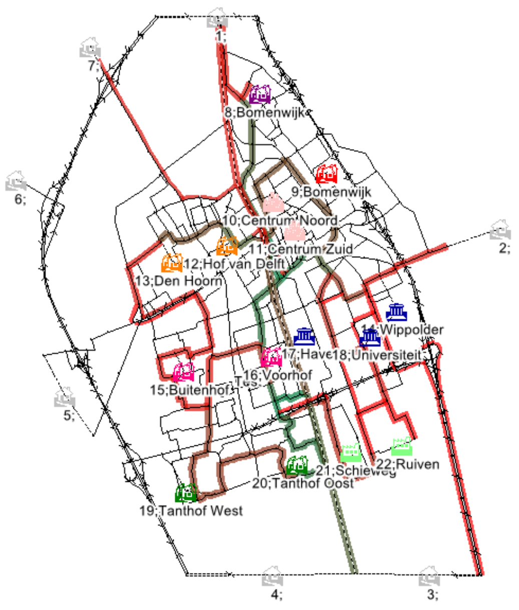
For comparing the different solutions also a look was taken at the route distributions based on the found OD matrix. The figures coming from these distributions are shown below:



(a) Route distribution of the real OD matrix

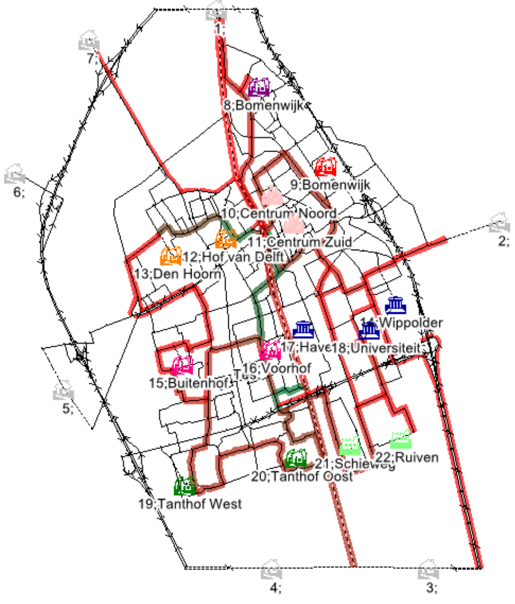


(b) Route distribution from COUNTS

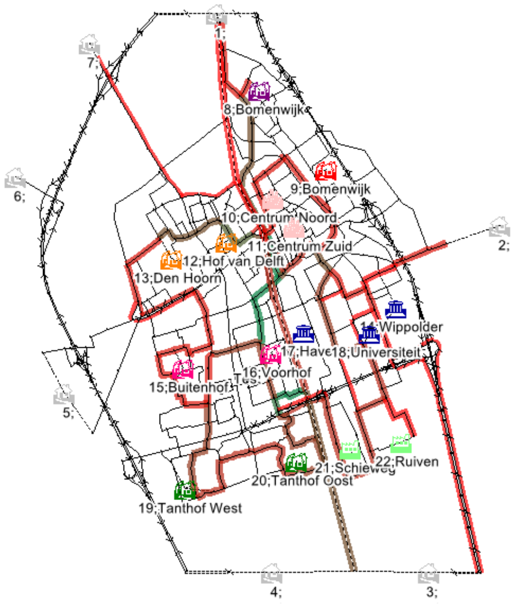


(c) Route distribution from STOPS

Figure C.2: Route distributions for the Delft network (1)



(d) Route distribution from TRACKS



(e) Route distribution from ALL

Figure C.3: Route distributions for the Delft network (2)

D. Test Networks Data - GSM

D.1 Simple Network

The GSM datasets used in the tests on the Simple network:

userID	time	x	y	range
1	0	- 65	310	100
1	10	170	500	130
1	21	100	890	100
2	0	- 65	310	100
2	12	-150	740	100
2	19	100	890	100

(a) GSM Dataset 1

userID	time	x	y	range
1	0	- 65	310	100
1	335	170	500	130
1	563	100	890	100
2	0	- 65	310	100
2	364	-150	740	100
2	624	100	890	100

(b) GSM Dataset 2

Table D.1: Two GSM Datasets for the Simple network

D.2 Delft Network

The GSM datasets used for the tests on the Delft network:

userID	time	x	y	range
1	0	84327	444196	200
1	166	83411	444599	600
1	274	83477	445466	600
1	380	84080	445950	600
1	479	84558	447213	300
1	642	84636	448434	300
1	643	84233	448539	600
1	736	84626	449177	600

Table D.2: Standard GSM Dataset for the Delft network

userID	time	x	y	range
1	0	84327	444196	200
1	166	83411	444599	600
1	274	83477	445466	600
1	380	84080	445950	600
1	680	84080	445950	600
1	779	84558	447213	300
1	942	84636	448434	300
1	943	84233	448539	600
1	1036	84626	449177	600

(a) Dataset for test L1

userID	time	x	y	range
1	0	84327	444196	200
1	166	83411	444599	600
1	274	83477	445466	600
1	380	84080	445950	600
1	479	84558	447213	300
1	640	84233	448539	600
1	642	84636	448434	300
1	643	84233	448539	600
1	736	84626	449177	600

(b) Dataset for test L2(a)

userID	time	x	y	range
1	0	84327	444196	200
1	166	83411	444599	600
1	274	83477	445466	600
1	380	84080	445950	600
1	479	84558	447213	300
1	610	85315	447064	300
1	735	84558	447213	300
1	898	84636	448434	300
1	899	84233	448539	600
1	992	84626	449177	600

(c) Dataset for test L2(b)

userID	time	x	y	range
1	0	83132	444582	200
1	660	81905	449845	200

(d) Dataset for test L3(a)

userID	time	x	y	range
1	0	83132	444582	200
1	1080	86235	446797	200

(e) Dataset for test L3(b)

Table D.3: GSM Datasets for the Delft Network

From the different tests several routes were found. These routes are shown below:

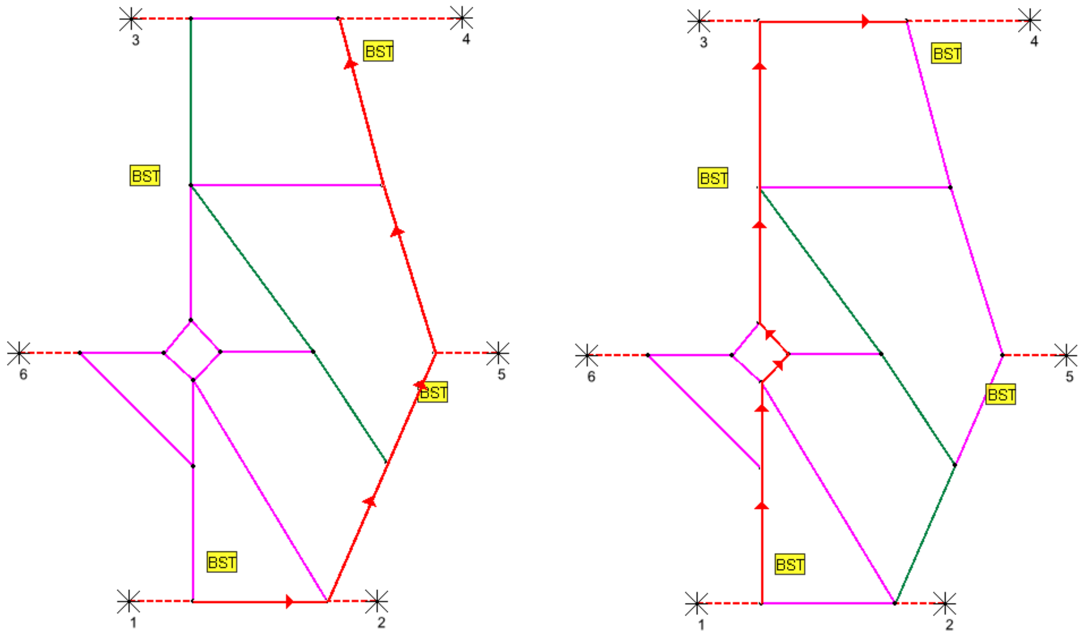
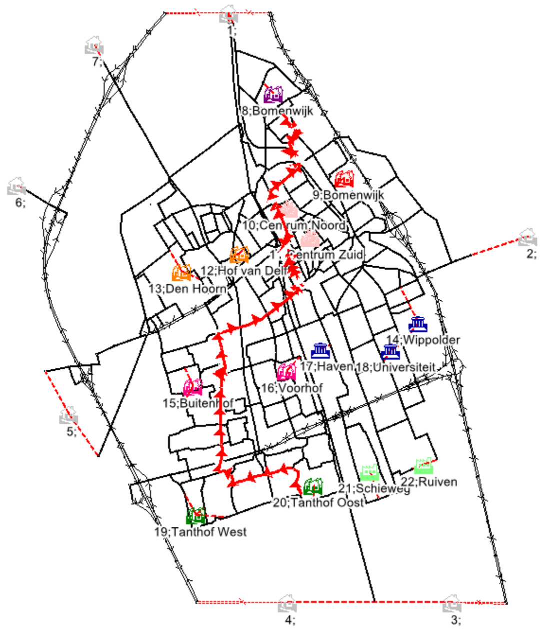
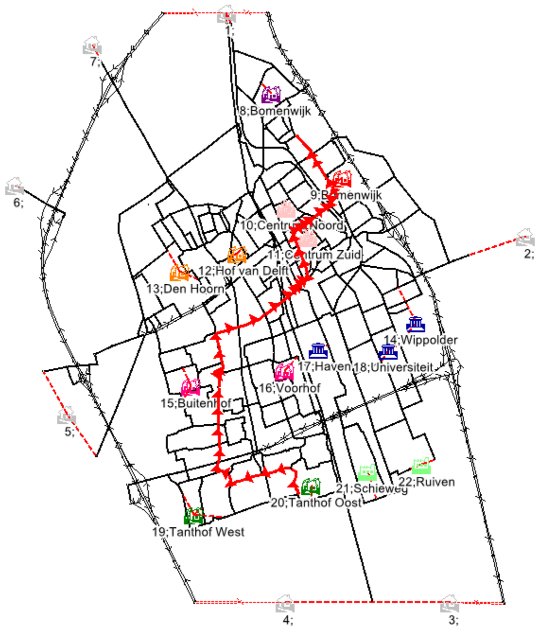


Figure D.1: Two routes for the Simple network

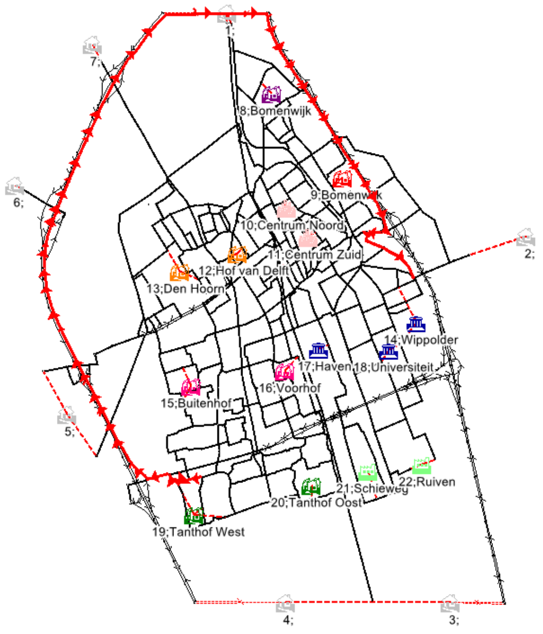


(a) Standard route for the tests

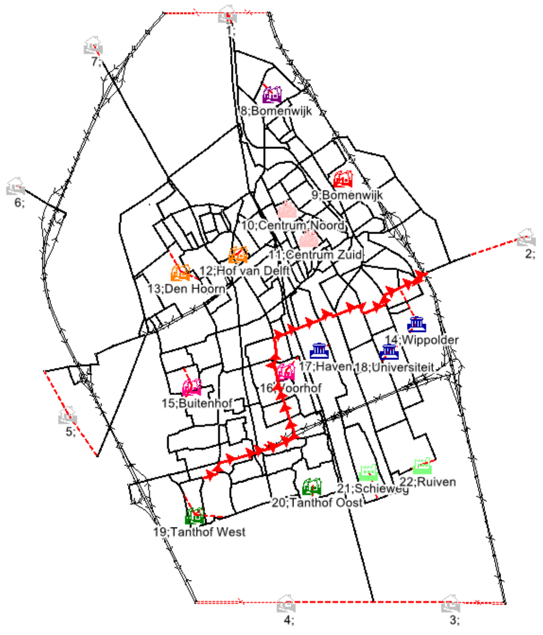


(b) Route found by test L2a

Figure D.2: Routes from the Delft network (1)



(c) Real route for test L3b



(d) Route found by test L3b

Figure D.3: Routes from the Delft network (2)