Personal Advanced Traveler Assistant

Dynamic Traffic Assignment

Ana Andreea Radu

Personal Advanced Traveler Assistant

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

MEDIA AND KNOWLEDGE ENGINEERING

by

Ana Andreea Radu
born in Braşov, Romania

Man-Machine Interaction Group
Faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS)
Delft, The Netherlands
www.ewi.tudelft.nl

Personal Advanced Traveler Assistant

Author: Ana Andreea Radu
Student id: 1391917
Email: radu.andreea30@gmail.com

Abstract

In spite of numerous road management schemes and developed infrastructure the society nowadays still faces the problem of highly congested roads due to the increasing traffic demand.

The focus of this thesis is to develop a complex and integrated system that addresses the challenges of dynamic traffic assignment in modern times. To reach this goal we had to study the theory behind such systems, come with new ideas and implement them into a real world human-centered traveler information system. Taking into account state of the art systems in this field we built a design for a Personal Advanced Traveler Assistant (PTA). The main purpose of PTA is to give routing advices depending on the users preferences and the available capacity in the network. The system incorporates a dynamic traffic algorithm that employs a prediction model. The prediction model should use current traffic updates and the routes of the guided cars in order to achieve an accurate prediction of the future traveling times.

Next step was to implement a prototype. This is available through multiple interfaces like smart phones or desktop applications.

The prediction model that the algorithm uses is based on historical data. Given the time constraints the prototype uses now only historical data but it can be easily extended to include live updates. This allowed us to build a prediction model that considers typical traffic dynamics in order to estimate the traveling time.

The performance measure of the traffic assignment algorithm is the shortest traveling time. The algorithm that we implemented is a time dimensional-extended version of the Dijkstra algorithm. The algorithm is tested on real data collected from the highways in The Netherlands. We evaluate the performance of the algorithm by comparing it to two versions of the static Dijkstra algorithm.

To conclude, we successfully implemented a working prototype that uses various technologies such as Java, the Open Street Map API for rendering the map or J2ME for the mobile phone client. The prototype that we have built represents a working proof of concept for a dynamic routing assistant. One advantage of the structure that we chose to implement is that each component can be further on extended independently. In this way we showed that such a system is feasible but we also left the possibility for different parts of the system to be extended into more advanced applications.

Thesis Committee:

Supervisor:	Prof. Dr. Drs. L.J.M. Rothkrantz, Faculty EEMCS, TU Delft
Committee Member:	Dr. Ir. P. Wiggers, Faculty EEMCS, TU Delft
Committee Member:	Ir. H.J.A.M. Geers, Faculty EEMCS, TU Delft
Committee Member:	Dr. Ir. D. Datcu, The Netherlands Defence Academy, TU Delft

Preface

The master thesis that you are reading represents my graduation project as a Master student at Delft University of Technology. I would like to take this opportunity and thank those who helped and supported me during the last 2 years. I thank especially to Prof. L.J.M. Rothkrantz for guiding my steps to successfully accomplish this master thesis. I am grateful to him for all the support he offered me, the efforts he invested in this research project, for giving me new ideas whenever I though I reached a dead end and not only. I would also like to thank him for trusting me and for making me feel more confident. I consider myself lucky to have met him and especially for having him the supervisor of my master thesis.

Special thanks go also to my family, my mother, my father, my sister and my grandparents who always supported and encouraged me to follow my dreams. In spite of the distance between us they always helped me to remain optimistic.

Because work seems easier when you are surrounded by friends I use this occasion to thank all my friends I met during my stay in The Netherlands, for their advices and for the nice moments that we spent together. Special thanks go to my collegues and to the teachers from the MMI department but also to the people from the lab, for all the good times and for the Dutch lessons, of course. In this way I would also like to show my appreciation for all the Romanian students at TUDelft.

Finally, I thank Vlad Mihai Sima for supporting and always encouraging me that everything can be achieved if you invest the effort. I am thankful for all the hints and advices that he gave me during these 2 years and for keeping me optimistic.

Ana Andreea Radu
Delft, The Netherlands
August 17, 2010

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every morning commuters and drivers traveling in urban areas all over the world suffer from delays due to congestion in traffic. This leads to increased costs as vehicles consume more fuel and people arrive too late at the working place. So, they are not able to use their time efficiently. They generally try to minimize their traveling costs as much as possible. Nevertheless, the repercussions of congestions are firstly economical but also environmental.

In spite of well-planned road management schemes, developed infrastructures, modern tracking technologies and traffic rules for safe driving, society nowadays faces the problem of traffic congestion due to the ever increasing traffic demand. Building new roads can be a solution but not a feasible one due to costs and environmental concerns.

Especially nowadays, time is an important resource. Not only do people lose much time because of congestions but they also get angry or frustrated because they are late for an important meeting, at their job or other events in their lives. Let us follow a short story in the following paragraph that could have a different outcome.

Imagine that you just finished a business meeting and you have to arrive at another one at 10 o'clock. You see that your GPS suggests that you will arrive there in 40 minutes and that is plausible as long as that is not a peak hour. However, you hurry up and leave one hour before. Everything goes fine and after 35 minutes you see that there are only 15 more km to go. You exit the highway and you notice that there is a temporally closed road on your route. You turn and try to reach your destination somehow although the GPS insists you go on the same closed road as it is seen to be the shortest path. After you manage to "persuade" it to choose another route by getting further away from that point, you already see that are only 20 minutes left to the meeting. However, probably enough time for a few kilometers and finding a park place. Next the GPS chooses an arterial road for a couple of kilometers but there was an accident. You cannot do anything else than wait and feel angry and frustrated because this could have been avoided. You feel unlucky, but still hope for a miracle to happen and get to the meeting. Finally, you get all sweaty to the meeting being 40 minutes too late. You apologize for being late while the others assure you that they are also sorry but they bought the product from the competitor of your company.

1.1 Problem definition

The Netherlands has around 16.6 million inhabitants and in January 2003 there were 8.3 million vehicles registered and more than 250 million vehicle-kilometers each day across its entire highway network. It is a densely populated country with almost 200 cars per square km.

In July 2007 the Federal Highway Administration and National Cooperative Highway Research Program in US sponsored a scanning study to examine congestion management programs and policies in Europe [1]. This program included four countries: Denmark, England, The Netherlands and Germany. The focus was on the traffic parameters and on the measures for the deployment of congestion. Figure 1.1 [1] shows a graph from this study which indicates the traffic growth in The Netherlands in the last fifty years with a continuing trend of about 3 percent per year. In

order to show some statistics, The Netherlands reports congestion costs of 0.8 M€per year. The accessibility of main ports is also affected [2]. The threads cited in the same official report refer to the distribution function of traffic, jobs and economy, environment and safety. This shows a critical need for traffic congestion management.

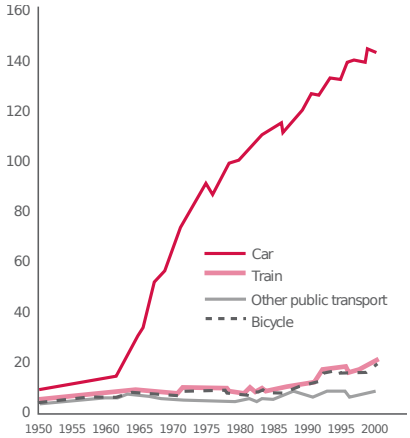


Figure 1.1: Fifty years of traffic growth by mode in The Netherlands

In The Netherlands traffic management measures started to be taken already from 1989 [1, 3]. The traffic policy relies heavily on traffic management. Only building new roads would be too expensive or difficult because of spatial and environment conditions. Therefore, the efforts were turned towards traffic management. From 1989 onwards, various traffic management measures were implemented, starting from motorway traffic management systems to overtaking prohibitions for trucks and special police teams for rush hours. The efforts undertaken so far to improve the existing traffic network through traffic management schemes by the government in The Netherlands are the following (see Figure 1.2 [3]):

- Queue warning.
- Speed Harmonization.
- Temporary Shoulder Lane use.
- Dynamic Lane Marking.

Besides these measures that are applied on the roads in order to alleviate congestions, an alternative is to make more efficient use of the infrastructure. This has lead to a great development in the area of *personal intelligent traveling systems*. These systems are researched in order to improve the traffic parameters.

How congested are the arterial roads at peak hours? What is the travel time to certain destinations at different times of the day? Which are the chances to run into a highly congested road? Which are the factors that determine traffic congestions? All these questions arise every day in the minds of millions. Because of this it is hard to estimate when people should begin their journey in order to arrive at the destination in due time and without waiting too much in traffic.

The dynamic nature of traffic can be observed in both temporal and spatial changes in the traffic demand, roadways capacity and traffic control parameters. The traffic demand increases over time until peak periods, it varies stochastic during the peak hours and decays at the end of peaks.

A great part of the congestions are regular and recurring at certain locations of the freeway network and also at certain hours. These add up with the incidents, the road maintenance fields, the weather conditions and special events that influence traffic.

Figure 1.2: Examples of traffic management measures in The Netherlands

Most GPS devices that people own, when requested a route, provide a solution based on the minimum distance. People usually hear on the radio if there is an incident or if there is something special happening on the motorways. But then they have to estimate themselves which alternative route they should choose, how much extra time they have to spend, whether a certain alternative is the most convenient one and so on.

We conclude that congestions can influence our lives in many ways. And even if there were a lot of approaches meant to bring an equilibrium in the traffic or to search an optimum for each individual driver (by minimizing his costs) there is a great need for more improvements.

1.2 Dynamic Traffic Assignment

How can we best define the dynamic traffic assignment? It can be described by the process of dynamically selecting a path made out of roadway segments from a trip origin to a trip destination depending on a cost function. The cost function is usually the traveling time.

The complexity of traffic management is due to the interaction of three main processes [4]:

- The traveler's decision behavior, as the decisions of drivers influence the outcome of the traffic network;
- The dynamic traffic assignment in a traffic network;
- The traffic flow behavior, in particular when incidents and accidents occur in the network.

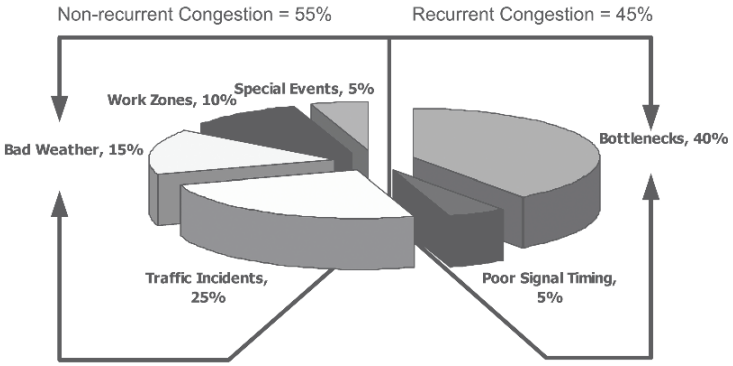


Figure 1.3: Causes of congestion

All these traffic factors interact with each other and change the traffic flow behavior and the traffic assignment. The empirical traffic patterns and their predictions are an interesting field of complex systems because of all these change. The physics of traffic flows with all these processes are a great challenge for the *traffic prediction* approaches. There are recurrent patterns that can be noticed in the traffic data, not only for peak hours but also for the days of the week. There are usually great differences between Sundays and Mondays for example.

Furthermore, incidents are a frequent source of disturbance in the traffic but there are more factors leading to congestions such as the *weather conditions, the drivers behavior, special events* (such as Christmas or an important football match), the *road maintenance fields* etc.

Figure 1.3 [1] shows a classification example of the *non-recurrent (55%)* and *recurrent congestion (45%)* in the United States. For the first one traffic incidents have the greatest contribution (25%) followed up by bad weather conditions (15%), work zones (10%) and special events (5%). For the recurrent congestions bottlenecks have the greatest contribution (40%) followed up by poor signaling time (5%). This classification indicates that long term predictions based on historical data and short term predictions that are based on current updates from traffic have to complement each other in order to succeed to correctly estimate the traveling time.

Traffic predictions are important for the drivers as they would be able to realistically asses the time they need to get to the destination. Moreover, correct traffic predictions may also be used for safety and also to improve intelligent applications of the vehicles, such as the adaptive cruise control.

1.3 Personal advanced Traveler Assistant (PTA)

Figure 1.4: Use of a personal traveler assistant

One approach to solve the congestion issues and though to reduce the travel time for individuals is to develop a route planner that incorporates current and future traffic information when searching for the best route. When congestions or incidents occur on this route the planner has to compute the best alternative solution which may lead the driver on different roads or to a train station.

The main purpose is to minimize the traveling time by taking into account the changes and the future situation in the traffic network. In this area the personal advanced traveling assistants play a crucial role. Such an assistant has multiple functions including that it computes the shortest traveling time-routes based on current information received from traffic.

Within the research program "Seamless Multimodal Mobility" at Delft University a Personal Intelligent Traveler Assistant (PITA) has been designed. A more detailed description of PITA is presented in Chapter 3, Section 3.1.

Our purpose is to continue the work from PITA and develop our own dynamic traffic assignment integrated in a system that will be called Personal advanced Traveling Assistant (PTA). In the remaining of this section we will synthesize the main features of PTA.

PTA is a *smart traveling assistant* that gives the user routing advices during his trip. PTA benefits of current updates of the traffic states. This means that it suggests the best route at departure and it offers alternative solutions in case of congestion or unexpected events. PTA is smart because it is able to manage traffic information, analyze different possibilities and conclude which is the best alternative route. Moreover, when it chooses a route it takes into account the driver's preferences.

PTA connects to the user through a hand held device, such as in Figure 1.4. The mobile phone stands as an interface between the system and the driver. It connects to the network in order to obtain the necessary information.

PTA also benefits of a smart interface: it is able to offer the user different input and output data modalities depending on the situation. Thus, it has a multi modal user interface that may include speech, speech recognition and a graphical interface.

1.4 PTA - travel server system

PTA represents a distributed system that connects the main server to the users. The clients of the server can be either desktop computers or mobile devices.

We may assume that in the next future many/all car drivers will have a hand held device/routing device. It can be either a TomTom-like or a smart phone with an implemented routing system (GPS).

As aforementioned, we design a system which is able to route car drivers taking as input the origin, the destination and the departure time. Usually the origin of the trip is known by the GPS receiver in the device. The system provides a solution and *supervises* the drivers.

During the trip, at regular time intervals, the car driver provides his position using the GPS device or a smart phone (this is actually done automatically). In this way the system will get real time information on the current traffic flow. This information will be used to update the routing tables on the server.

At this moment we assume that the routing software has to be implemented on a central server and that every hand held device has regular contact to the server. These devices represent clients of the central servers. The software for the clients could be downloaded but the dynamic traffic information would be available just at central points, and could be also downloaded. But this is currently beyond the network capacity and computational capacity of the hand held devices.

Travel time update

For our routing algorithm we start with a database of historical traffic data. This means that we know the travel time t_{N_j} on a segment N with the time intervals $i_{N_j}, j \in (0:00, 24:00)$. The important part is that as soon as real time information of tracked travelers becomes available, the database may be filled with additional new data, so that we update all travel times t_{N_j}. Because we know the position of the vehicles at regular time intervals and their destination, we can compute current time t'_N along a segment N. Then the updated travel time $t_{N_j new}$ can be computed as follows (Equation 1.1):

$$t_{N_j new} = t_{N_j old} + w\dot{t}_N. \tag{1.1}$$

In this way we have a real time updated database of travel times. But the main problem is not to have real time travel information but also reliable predictions of future. In our case, we have the destination as input data and our system provides a travel advice. Let us assume that the traveler takes this advice seriously and then we know which is the expected travel stream in the future for a representative group of travelers using our PTA system. Given the historical travel

stream we are able to make predictions about the relation between future travel streams and the corresponding travel times.

So, we could train an ANN with the relation between the travel stream and the travel time. Given predicted travel stream density we can compute future travel times which will be continuously adjusted by as real time traffic information becomes available.

To conclude, we are able to compute the time of trips/routes not only based on current real time information but also based on reliable predictions of future travel time.

In this thesis only part of this challenging approach could be implemented and tested.

1.5 Scientific challenges

Smart traveling assistants aim to improve the traffic flow. Our challenge is to build an integrated dynamic traffic planner for individual travelers. An important component of this planner is a traffic prediction model.

Our traffic assignment model will use a cost function determined by an established performance measure. The *performance measures* considered in literature [5] are the *traveling time*, the *fuel consumption*, the *price of the trip*, the *safety on the roadways*, the *reliability of the guidance system*, the *robustness of the guidance system* etc. There are many parameters that have to be considered. The driver's behavior can be another example. Most drivers are interested in the shortest traveling time whereas others are interested in the smallest toll charge. However, there are also other aspects related to drivers: if some wish to drive with a low constant speed in order to consume fewer fuel, they might delay the others. And in case of the zipper effect many drivers may not be fair, overtake others and then place in front. Fuel consumption and congestions also influence the traffic flow. Moreover, incidents, unexpected events or weather forecasts are other parameters to be considered.

How could we approximate a model that considers all these variables? One possibility is to resolve a regression model (Equation 1.2) for the traffic flow.

$$M_i = \beta_1 x_{i1} + ... + \beta_i x_{ip} + \varepsilon_i = \sum x_i \beta + \varepsilon_i, \tag{1.2}$$

where M represents the measured variable, namely the expected cost of the trip model and x_i stand for the whole multitude of input variables. We realize that the traffic model that we have to achieve is a multi-parameter problem. This represented a scientific challenge for the research in the field of traffic assignment (Section 2.3).

The performance measure that we consider for our prediction model is to minimize the traveling time. Further on, the traffic prediction model has to be integrated in a smart traveling system. First, we have to establish a design of a smart traveler assistant and then implement a prototype.

Designing a smart traveler assistant. Our first target is to design a smart traveler assistant. This involves various factors. A prediction model for the traveling time and a dynamic traffic assignment algorithm have to be incorporated in the system. The estimation of the traveling time should be done based on current traffic information and prediction models. The system is developed in a network and it relates to the user through a hand held device. The data processing and the route computation are done on a server that is connected to the users devices. The server is linked to the traffic network database and to the incidents database.

We aim to design a multi modal route planner that continuously benefits of real life traffic data. First, we need to figure out how to model a multi modal dialog manager in the network between server, users, traffic data, incidents and even other means of transport. Further on, we also attempt to have a human-centered design. People usually have various preferences and the target users of such a system are very diverse. Based on the preferences PTA assigns each user a profile. In order to detect important aspects related to the interface or the routing preferences a user test has to be done. However, it is not trivial to establish which are the key factors for a human-centered design with regard to the PTA.

Build a traffic model. It is a challenge to develop a dynamic traffic assignment model that describes as close as possible the real life traffic situation. The model should estimate the traveling time with a low prediction error. First, it is debatable what would be ideal: a model which guides individual travelers or groups of individuals? As it is quite complex to have a network in which groups can be guided we will resume at routing individuals. In this way we aim to assign each user the best possible route. But how does this affect the other participants at the traffic?

Furthermore, the behavior of the traffic flow and the traffic network have to be introduced in our model. Developing a model that stands for the real one is a great challenge for researchers in this field. Why is it generally hard to simulate/test a traffic model from real life? Because for example, not all cars are controlled by the same system. Some drivers might even use paper maps, some might rely on their own (incomplete) knowledge or use GPS devices that do not belong to a network. Because of such issues traffic models in research come along with assumptions.

The approach for our traffic model is influenced by the assumptions that we established. For the implementation of our system we assume roads to be straight lines. For the highways of The Netherlands this assumption will not influence much the outcomes. The length of the vehicles will not be considered because we implement the routing at a "macro-level". This means that we will not concentrate on intersections, the number of lanes or the capacity of each highway but we will focus on the relationship between the traffic flow, density and speed.

The algorithm. Our traffic model needs a simple and scalable algorithm that assigns the shortest path in time for a trip. In other words, the algorithm has to solve the dynamic traffic assignment problem. An approach for solving this is to divide the time horizon in steady-state time intervals and apply a shortest path algorithm to each time interval depending on the average speed of the vehicles.

Testing the algorithm in real life would be very hard at this moment because neither can't we access the devices for tracking nor can't we route real individuals. Therefore, the algorithm will be tested in a simulated environment and a comparison to a static traffic assignment will be done and assessed.

Prediction models. Traffic prediction models demand a good understanding of the traffic patterns and their occurrence in order to achieve a low prediction error. Traffic predictions use historical information on traffic variables. Due to the progress done in the traffic data measurement and acquisition the variety and complexity of the available data is increasing. There is a lot of acquired data and it is complex to handle all this data in an efficient manner.

Our system needs reliable data to make long term predictions. The prediction model of our prototype is based on historical data. However, a reconstruction of the current traffic state for short term predictions is considered. Incidents or special events which change the current state of the traffic network are also important. Incidents can be of many types and may have various outcomes. An advanced traveler assistant should first detect the type of the incident and then manage the situation. In our model we aim to develop an incident management model. The delays prediction in case of incidents is a state of the art research subject. The decrease in speed and the length of the queue that forms have to be studied. The delays determined also have to be estimated. Then, users who were assigned that road (or an affected one) have to be re-routed. But a balance should be maintained between the roads in the neighborhood. Cars cannot be all redirected to another single road close to the closed one. But how to manage a redistribution of the coming traffic in such cases?

The management of incidents assumes a close inspection of the number of affected lanes, current traffic, the distance between the vehicles, drivers behavior, the time that authorities need to remove the incident and so on. However, it is difficult to manage incidents without a method that takes into account lanes, the vehicles width or the driver behavior.

Implement an integrated system. The implementation of a prototype is not a scientific challenge but it is a challenge because it has to be an integrated system and it should use various

open source technologies for each component. An integrated system refers to a network application that can be accessed by the users. The users requests are sent through the network to the central server which sends back an answer. The interface between the system and the user is represented by a hand held device. Current traffic information should be processed by using the position of the vehicles which are already sent.

Integrating many different technologies made by different companies is a big challenge not addressed today.

Traffic models should be also considered in the system. There can be various traffic models applied to each situation. These models can represent the days of the week, different weather forecasts, events or national celebrations.

Human-centered interface design. PTA has to be an interactive system that people find easy and enjoyable to use. Because of this, the usability of the system has to be taken into account at design and implementation. The questions that arise with respect to the design of a user-center interface are the following: which are the key factors for a human-centered interface of such a system? What should a multi modal interface comprise, how to display it and how to switch between different modalities?

Although we aim to run a user test with regard to these issues the diversity of the users and the lack of time or of a larger sample of subjects for tests the design and the interface of the PTA represent a challenge.

1.6 Goals of the thesis

Our goal is to develop a complex system to address the challenges of car routing in modern times as presented in Section 1.5. To reach this goal we have to study the theory behind such systems as abstract models and implement these ideas into a real world user oriented advanced traveler information system.

In the following paragraphs the main goals of this thesis are going to be presented.

Develop a traffic prediction model. One important step for the system is the development of a dynamic traffic assignment model. A crucial component of the model is the prediction of the traveling time. Figure 1.5 shows a scheme for the variables of the traffic prediction model. The main variables are the *historical time series* such as the *average speed* at particular times on segments of roads or the *travel time* along with the current status. These represent the input data for the traffic predictions [6].

With respect to the traffic model, we will consider a reduced model for the highways and cars. We will ignore the spatial characteristics of the car as length and width. We will also not consider lorries and the number of lanes. For highways we will capture the number of lanes by using just the abstract number of the average speed. So, we will not be concerned with the number of lanes, width of lanes or any other related issue. These assumptions of our traffic model will be explained and motivated.

Develop a routing algorithm based on the prediction model. The algorithm that we develop has to take into account the changes that occur in the network through time and not just to consider some given states. In order to build up the algorithm a non-equilibrium method is chosen. This assumes that traffic requests between two zones are assigned the path which has the minimum cost. Moreover, the purpose of the algorithm is to route individual users. The algorithm will be tested on a traffic network that is actually a replica of the highways network of The Netherlands.

Identify a metric with which we'll evaluate the algorithm. The underlying objective of the algorithm is to reduce the traveling time. The cost function in the algorithm is associated with the traveling time. Minimum path algorithms include the models developed by Danzig and Dijkstra. Therefore, the proposed algorithm is a dynamical adapted version of the minimum path

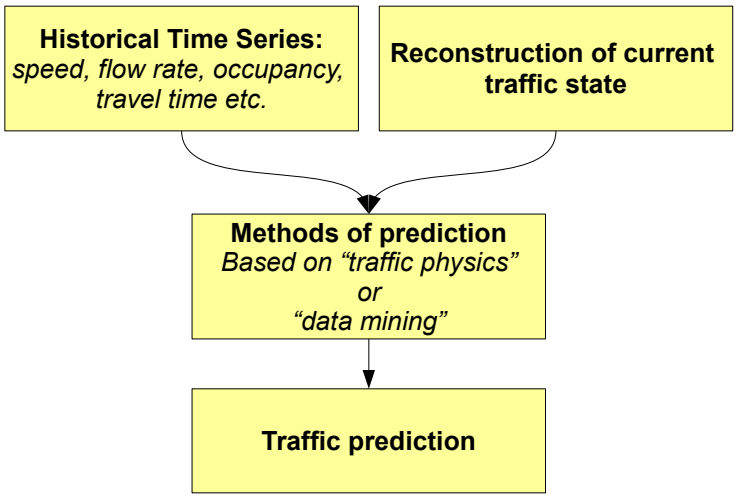


Figure 1.5: Variables scheme for traffic prediction

assignment of Dijkstra - Dynamic Dijkstra Traffic Assignment (DDTA). DDTA is required to provide a trip request with the shortest route in time. But the traffic states change during the day and this means that we deal with a varying speed during a day. In order to integrate the variations of the traffic, DDTA would use for the moment historical data collected from the highways.

The algorithm will be tested by comparing it to the static minimum path Dijkstra algorithm. Given the same input data, the resulted traveling times will be compared and analyzed.

Build a design for a personal advanced traveler assistant (PTA). Figure 1.6 synthesizes the main components of an advanced traveling information system. An important part is the DDTA algorithm which computes the travel time and predicts the next state of the network over a short period of time. The DDTA needs as input data the route demand and the historical data which contains the traffic flow-average speed relation. The algorithm connects to the databases in order to access the traffic network, the historical data, information on the current traffic or the incidents. The user interface plays as well a crucial role in such a system. The design which we propose for the PTA is a human-centered approach. An important thing is that we need a system that is easy to learn, easy to use and user friendly. PTA will link to the users by using mobile phones but also a desktop application. The mobile phones will be used also as tracking devices. Although this may rise a privacy issue, it is not the case because we intend to keep track of the cars by simply using ids in order to compute the average speed-traffic flow relationship at different points on the highways. This information contributes at the traffic predictions on the short term.

Briefly, the system that we design has the following requirements :

- The first requirement is the need of an integrated system that allows running the algorithm separately and connecting to the users through a hand held device and a desktop application;
- Design and implement a network which links each component of the system. The algorithm will work on a server to which both the mobile and the desktop client are connected;
- Design a multi modal user oriented interface;
- Implement an interactive graphical user friendly interface.

Implement a prototype. Following the design of PTA our next goal is to implement a prototype that has various functions. Why do we implement this prototype? Because this can be a testbench for algorithms and multi modal interfaces. We need an open system that allows

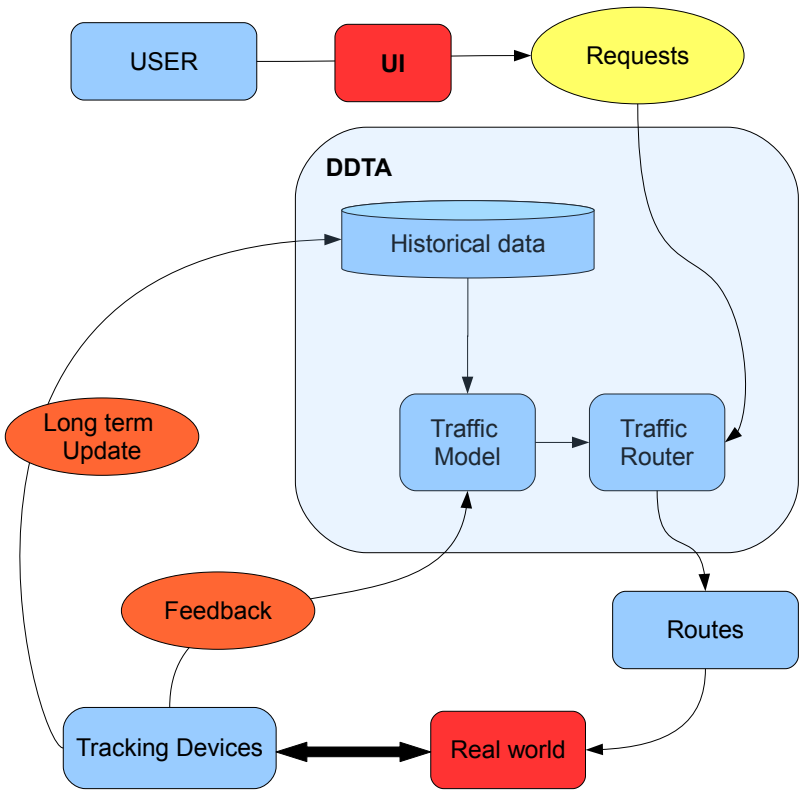


Figure 1.6: Schema of an advanced traveling information system

simulations and tests. Thus, the purpose is to provide a workbench. For the prototype each main component of the proposed architecture has to be implemented.

The server will connect to a database that provides historical traffic data in order to compute the travel time. The system will be modeled in such a way that it does not depend on the type of the data. First, historical data will be considered in order to develop and test the prototype. Next, it will be updated based on the live traffic information. In this way, different traffic models can be also formed: for different types of day, forecasts or events. However, we aim to have a system that allows a continuously updated database.

In the following the most important goals for each component of the system will be briefly emphasized.

Due to the requirements aforementioned our system has to rely on a *client-server* communication. The server receives a route request, connects to the database (to retrieve the roads, the intersections and the historical data), applies the algorithm and returns the minimum path along with other details. The server connects to a mobile and to a desktop client. The client-server approach fulfills the requirements on modularity (client and server, different modules) and scalability (we can add multiple servers for example).

The UI has to be a user friendly interface that enables the user to learn and use the system quickly. It has to offer relevant and important information. With regard to the UI of the desktop client, this should also allow administrators of the system edit the data from the database.

The UI of PTA should be attractive also to non-technical users as the target group of a GPS application is really diverse. Because the design envisions a user-centered application the interface of the mobile phone and the interface of the desktop application will be developed after analyzing a user survey. This involves running a survey among possible subjects in order to detect what is

actually important to people. So, the purpose of the survey is to emphasize aspects that we should include in the UI of PTA. However, due to time and computational constraints we aim to make a selection of these. Nevertheless, we will motivate the reasons of our choices.

As mentioned already, the mobile client of the system is represented by a mobile phone. Our goal is to implement an application on the mobile phone which is able to send the route requests from the user to the server. Next, receive the route that minimizes the traveling time along with the other details. So, the mobile phone has to send the departure, the destination point and the departure time to the server. The GUI of the mobile phone provides the user an interactive map on which the route will be drawn. Due to time constraints and to the novelty of programming a mobile device the GUI will display the map, the roads, the roads, the route and will benefit of one modality to make a request. The GUI of the desktop and mobile device have to be both eventually tested in order to get feedback from potential users of the system.

Generic requirements of the system. For the system that we implement we have requirements which are generic for a project of this complexity and size. These are the following:

- *Portability*: the software of the application has to be portable; this feature allows moving the application to different environments without modifying the code/structure.
- *Scalability*: it should not matter that the available data is just a subset of a larger dataset which might include all the roads in The Netherlands. The architecture and the algorithm have to be designed in such a way that replacing the current data source with real life data will be an easy task.
- *Expandability*: it should be easy to expand the system, to incorporate new information, new input or output modalities, or ways of getting the data.
- *Modularity*: all the modules should have clear interfaces and all modules should be independent.

Evaluate the design, the implementation and motivate our choices. Besides the scientific challenges of the research, our methodology and the dynamic traffic assignment description, part of this thesis is dedicated to the design of PTA and the description of the architecture that we chose to implement. Our prototype aims to prove the concept and develop a conducive environment for test research.

Equal efforts are invested in the scientific sphere of this thesis as in the implementation of the application.

Test the algorithm on real data. Testing the system on real data is important in order to validate the algorithm of the application. The algorithm will use real data collected from ANWB for all highways in The Netherlands along with some national roads. This data can be used with various purposes but it is needed for the prediction of the traveling time.

The distances between nodes on the map are also real. In this way, we wish to compare clear situations that people experiment every day.

1.7 Methodology

First, a literature survey was conducted starting with research papers, journals, reviews and ending with state of the art traffic simulations and designed intelligent traveling assistant systems. This helped to better understand traffic assignment models and to get more insight in GPS devices features.

Because our first target is to build a design for PTA we had to get acquainted with state of the art designed or tested industrial and academic systems (presented at Section 3). We did

study various types of current traffic data as well. This resulted in a thorough inspection of traffic management schemes in The Netherlands (presented in Section 3.5).

Next we developed the design of PTA that includes a multi modal user-centered interface. This signifies that the user can interact in various ways with the system. Depending on the situation for example he might use a graphical interface or the speech recognition. There is a great diversity of users and not all of them prefer the same interaction modality with a device. Or not all interaction modalities fit to any situation. In order to design the system and the multi-modal interface we evaluated potential users of the system. Further on, we read about new systems, ideas, applications and state of the art approaches and models that aim at a human-centered design. And all these influenced our design.

When the design of the system was achieved we focused on the implementation of a prototype that was based on a client-server network. The algorithm that we implemented for the prototype is the DDTA (**D**ynamic **D**ijkstra **T**raffic **A**ssignment) algorithm. Initially the system used random data for simulating the speed variance during a day. Next, historical data taken from ANWB was incorporated in the prototype for more realistic traffic predictions. The results of the algorithm were analyzed based on the comparison to the static Dijkstra algorithm. With this purpose we had a static Dijkstra solution built on shortest distance and one on shortest time (assuming the general maximum speeds on the roads).

Finally, the considered extensions of the prototype include traffic models for various days/situations created by using current data reported from the mobile phones. Delays predictions in case of incidents along with re-routing the vehicles should also be included.

A summary of our actions along with the main goals is displayed in Table 1.1.

1.8 Economical and societal relevance

Building a personal intelligent traveling assistant for people has first of all many societal and economical reasons. In this section we will point part of the economical reasons for using a traveler assistant and the improvements in people's lives that it might bring.

PTA is aiming at the individual traveler benefit. PTA always returns the best route for the traveler and in this case an issue that is difficult to evaluate is whether it may happen that a user is favored on the expenses of another one. But, when each vehicle benefits of the ideal route it means that it cannot choose a better option. The system may address the benefit of individuals but the model assumes a permanent knowledge of the traffic flow in the network. Thus, theoretically, it cannot happen that some people are disadvantaged.

1.8.1 Economical

In the society nowadays there are multiple costs associated with congestions in traffic. The first one to mention is the time delay which can lead to multiple losses/problems for most of the travelers. Some of them might not be able to reach their job/meeting in due time. And this usually causes people a lot of stress. In other circumstances, people can lose half a free day in the weekend because of an incident on the highway instead of reaching a nice place that they'd enjoy.

There are also lorries delivering special merchandise that expires in a few days. Shipments of flowers or milk products usually need to be fast delivered. Although they have different traveling rules, lorries are a serious problem as the largest part of the merchandise transport is done on the highways.

Another issue is that congestions lead to increased fuel consumption and even more, increased volume of emissions from vehicles. Under congested conditions, there are two additional influences on the vehicle fuel consumption [7]:

- there will be a change in the vehicle speed which affects the aerodynamic and rolling resistance fuel consumptions;
- there will be decelerations and accelerations, which influence the *inertial fuel consumption.*

Table 1.1: Methodology summary

	Action	Goal
1.	Literature survey	Get acquainted with state of the art traffic models, systems and developed projects in the academic environment and industry.
2.	User survey	Get more insight in users preferences with regard to the multi modal UI, routing and incidents management.
3.	Design the PTA	Design an intelligent personal traveler assistant.
4.	Develop a dynamic traffic assignment model	Define a traffic model that would underpin our traveling assistant. We developed various models, by augmenting each of them with more and more advanced processings. The last model is the most advanced.
5.	Study the available traffic data	Get acquainted with traffic measurements in The Netherlands, existing traffic data and databases, study the traffic parameters and eventually integrate a historical traffic database into the prototype.
6.	Implement a prototype	Develop a client-server application. Provide a workbench for algorithms and multi modal interfaces; an open system that allows simulations and tests.
7.	Test the algorithm	Analyze a comparison to other algorithm with regard to the traveling time, distance, most relevant differences and fuel consumption.
8.	Insert incidents in the traffic network and test the algorithm	The same comparisons should be analyzed but on routes where incidents occur.
9.	Test the system	Test the system in different scenarios.

The most significant component of the two is the inertial fuel consumption, particularly for heavy vehicles or those with high acceleration rates.

The vehicle operating costs are also increased and here we can think of tires and other components.

If all these costs decrease it is possible to *increase the trade* and even *decrease prices*. A continuous increase of the roads number in the infrastructure would alleviate these problems. As building new roads is not always feasible, a cost-effective method to improve the situation that we have nowadays would be the implementation of real-time traffic prediction systems. Systems dedicated to travelers for both pre-trip plannings or en-route usage would improve the vehicles distribution in the network.

1.8.2 Societal

The reduction of congestions leads to a stronger economy because of improved accessibility. It would also enable a traffic and transport growth.

Moreover, if we consider the facts aforementioned we can also conclude that if people would all use an advanced traveler information system they can be eventually happier and more efficient. The daily stress load would be diminished. Less congestions also lead to less pollution and this means that in time we would live in a healthier environment. Especially for The Netherlands which has a very dense population and more than 250 million vehicle-kilometers each day across its entire highway network (as we already stated in Chapter 1). Commuter traffic to and from major cities

such as Amsterdam, Rotterdam, The Hague, Utrecht, Eindhoven, Zwolle, Enschede and Groningen causes congestion. Congestion in The Netherlands is difficult to resolve because there is rather little room for expansion. Proposals for a "pay per distance traveled" is thought to discourage car driving but has not been implemented yet due to car owner resistance. This measure would bring some advantages but it doesn't aim at the individual's preferences or conform.

1.9 Thesis Structure

The thesis is organized as follows. The problem definition along with our research goals and the societal relevance of the subject were stated in the Introduction chapter. The concepts of dynamic traffic assignment and personal advanced traveler assistant were also introduced to the reader in the same chapter.

The second chapter offers a review of the related work in the field. It starts by an introduction to traffic assignment methods and algorithms where the User Equilibrium and System Optimum assignments are presented. The traffic assignment algorithms developed in the last half century can be broadly divided into equilibrium and non-equilibrium methods. These are also described by referring different journals in which researchers developed a certain method. The dynamic traffic algorithms differ from their static counterparts by using a third dimension, namely the temporal dimension. The most popular approaches for solving the dynamic traffic assignment problem are the analytical-based approach model and the simulations. Literature within this research area is extensive. Nevertheless, we studied the approaches of various researchers for both modalities and present their novelties or drawbacks. The estimation of the traveling time is a very important part in dynamic traffic assignment. That is why we dedicate it a section that studies the factors which influence it, the types of measurements and eventually methodologies of forecasting it. The last subject discussed at this chapter is the effect of incidents on congestions. We end the chapter with a summary which discusses our conclusions on the literature survey.

We continue in chapter three with a presentation of state of the art industrial or academic products which relate to our research. We discuss the PITA project developed at Delft University, at the Man-Machine Interaction department which represented the design for a personal advanced traveling assistant. Further on, intelligent vehicles traffic management frameworks are presented along with examples. We emphasize their advantages but also the impediments of their use in real life. We present as well a short description of the project developed by Nokia in collaboration with Berkely University: they aimed to use drivers with GPS equipped mobile phones for collecting real life data. This was a really new onset for the use of mobile phones with this purpose. We end the chapter by presenting the latest feature of TomTom - a Dutch product, that assumes a real life traffic service for a monthly cost. Traffic management schemes used in The Netherlands during the last 10 years are also briefly presented.

Once the main topics in transportation research were covered, it is time to focus more on the topic of our research. Chapter 4 discusses the resources of our project. It presents an introduction on historical traffic data and the main corporations in charge with collecting and analyzing it in The Netherlands. This represented an important part in our research as we used real historical data to predict the traveling time. The management of historical data along with the network is also presented.

The open source tools that we used to develop our application are stated in Chapter 5. We motivate our choices and present the main functionalities that we implemented by using them.

Further on, Chapter 6 describes the design that we propose for the Personal Traveler Assistant. We start by presenting the features of PTA and then we motivate the human-centered design of PTA. The environment of PTA and the concepts which underlie the system are also described. In order to detect the users preferences we did a user survey and we represent and discuss graphically the most relevant results. PTA is based on client-server communication in order to connect the users who can either use a desktop computer and a mobile phone to the dynamic routing algorithm. We assume that the interface to the user is separate from the processings of the algorithm and connection to databases. The model of the central server is presented through a UML class diagram.

The chapter ends with the description of both clients, desktop and mobile. Here we provide a mock-up of the user interface in both cases.

Until chapter 7 we designed a full featured system for PTA, taking into account all the needed features and functionalities. From Chapter 7 onwards we will focus on the subset of the system that we choose to implement given the time and data availability constraints. Thus, chapter 7 presents the proposed dynamic traffic assignment model which was developed in more steps. We created more models, by augmenting each model with more and more advanced processings. The last model is the most advanced one.

Moving on, the algorithm that we implemented is described in Chapter 8. We present the main characteristics and explain it on a synthetic network in order to give more insight. Next, we compare the algorithm with two versions of the static Dijkstra algorithm and analyze the results.

The most interesting parts from the implementation of PTA are presented in Chapter 9. Details about the integration of a database or the Open Street Map API in order to render the map are presented. Moreover, the client-server communication is described and the user interface for both clients are presented.

Our conclusions along with the proposed incidents management model which is considered future work are emphasized in Chapter 10. At the conclusions section we discuss the achievements that we had with regard to the goals we stated in the Introduction.

Chapter 2

Literature Survey

This chapter aims to synthesize important approaches in literature which discuss and give various solutions to traffic assignment problems. We start by discussing two important notions, User Equilibrium and System Optimum assignment. For these two we present and classify various assignment models in Sections 2.1.1 and 2.1.2. We present the advantages and disadvantages of each in the context of the referred work. One important development in the field was adding a third dimension to the assignment problems, the time. The methods which rely on this third dimension are presented in Section 2.2. They can be further divided in analytical methods (based on a complex mathematical model) and simulation models.

One important component of a traffic assignment algorithm is the travel time estimation. We dedicate Section 2.3 to these travel time estimation models, by discussing factors that influence them, how measurements are done and the methodologies for forecasting the travel time.

We conclude the chapter by giving some insights into incidents and how they are treated in such a complex system. This is usually done using a statistical model of incidents, so we present such a model in Section 2.4.2.

2.1 Traffic Assignment

Traffic assignment is defined as the problem of finding traffic flows given an origin-destination trip matrix and a set of costs associated to the links. One solution for this problem is either that the driver drives on the optimum path according to his preferences, known as the *User Equilibrium (UE)* assignment or alternatively the path that minimizes the overall network's traveling time, known as the *System Optimum (SO)* assignment. Various researchers proved that there is a difference between these approaches. The fact that a model assigns the best route for an individual does not imply that it ensures a global optimum. Handling the capacity of the network optimally is a big challenge. Both notions of UE and SO along with a famous example are presented in Section 2.1.1.

The traffic assignment algorithms developed in the last half century can be broadly divided into *non-equilibrium* and *equilibrium methods* which are described in Section 2.1.2 and 2.1.4. A classification of the non-equilibrium methods is presented in Section 2.1.3.

2.1.1 User Equilibrium (UE) vs. System Optimum (SO)

Wardrop [8] was the first one to differentiate the two methods. His first basic principle referred to the spread of trips over alternate routes due to congested conditions. He said that *"The journey times in all routes actually used are equal and less than those which would be experienced by a single vehicle on any unused route."* The traffic flows that satisfy this principle are usually referred to as "User Equilibrium" (UE) flows. The essence of this principle is that travelers would strive to find the path which minimizes their costs from origin to destination and the network equilibrium occurs when no traveler can decrease travel effort by shifting to a new path. These conditions are

called user optimal conditions, as no user will gain from changing travel paths once the system is in equilibrium.

His second principle said that *"At equilibrium the average journey time is minimum"* and this implies that each user behaves cooperatively in choosing his own route to ensure the most efficient use of the whole system. But this principle implicitly assumes that drivers perceive costs in the same manner.

A spectacular example that actually shows that the UE assignment is in general different from the SO solution is the Braess network. The mathematician Dietrich Braess obtained the paradoxical result that the addition of an arc to the network can result in increased origin to destination and overall travel cost. Caroline Fisk [9] studied the *Braess paradox* more in detail. She presented the sensitivity of travel costs to changes in the input flows while they are in a *Wardropian* equilibrium. Examples which state the fact that an increased capacity of the input flow can decrease the traveling time are presented.

Smith [10] reported another example of this phenomena. But he chooses a particular network with a certain shape and a reasonably natural mathematical model. He relates his research to network shapes that are very common, such as radials and two ring roads obtained by identifying all points on each ring. Even if the networks that were used for showing these concepts were small and simple, they could form parts of larger networks.

Steward [11] also emphasized three important aspects while using a simple two-link network and the example of Braess. First, the equilibrium flow does not necessarily minimize the total cost. This implies that sometimes it might be advantageous to restrict the use of a part of the network. Second, by adding a new arc to the network the total cost at equilibrium may increase. And this may also increase the equilibrium travel cost for each individual user.

2.1.2 Non-equilibrium traffic assignment methods

Non-equilibrium methods assign traffic to a single minimum path between two zones. The minimum path infers the minimum travel time. Minimum path algorithms include for example the models developed by Dantzig [12] and Dijkstra [13]. Other non-equilibrium methods include diversion models, multipath assignments and eventually combined methods.

The factors which influence the formulation of the road assignment models and the criteria by which different techniques may be compared and evaluated are presented by Dirck Van Vliet [14]. There are three basic parameters of networks and model characteristics that are referred, namely *the degree of mathematical convergence achieved by the network* (δ), *the degree of variations in the travel costs* (ε_1), *the relative travel cost average changes due to congestion* (ε_2) and the overall objectives. A specific model may be appropriate for a certain situation and inappropriate for a different case. Whether a model is suitable or not is expressed by using the parameters presented above. Different methods can be compared using ε_1 or ε_2. When a "cost-flow" relationship with a trip matrix is used, the total cost along each O-D route has to be computed. Nevertheless, the cost might differ among drivers: some might prefer the shortest or cheapest route while others the fastest if there are congestions. Thus, different sets of drivers "perceive" their travel costs differently.

The traffic model becomes very often a compromise between a large number of competing forces. Van Vliet [14] suggests a schematic presentation of the traffic assignment in order to illustrate the sort of decisions that need to be taken in order to set up an assignment model (Figure 2.1). The *input* includes the route demands (that is the trip matrix) and the network input (links, intersections, capacity, links length, number of lanes etc.). The definition of the generalized travel cost is also perceived as input. The *rules* refer to the set of principles by which drivers are presumed to select their route. The *procedure* refers to the mathematical model or the routing algorithm that uses the given input and satisfies the rules. The *output* consists of the link flows, route choices or turning movements given by the solution of the algorithm. The output should satisfy the overall *objectives* and requirements of the assignment model.

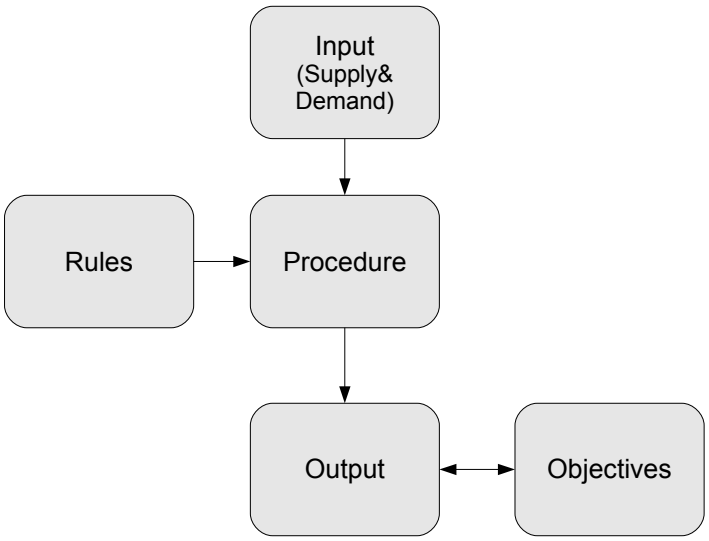


Figure 2.1: Schematic structure of the traffic assignment

2.1.3 Classification of non-equilibrium assignment models

Non-equilibrium models are classified into: (1) all-or-nothing assignments, (2) capacity restrained assignments, (3) diversion models, (5) multipath assignments and (6) combined models. They rather use non-optimal heuristics or other approximation solution techniques. Well known algorithms from the first category belong to Dijkstra [13] or Dantzig [12] for example. Their algorithms enjoy popularity and are preferred in engineering studies due to their simplicity and macroscopic nature.

These models make use of an *incremental traffic assignment technique* [15, 16]. The total traffic demand in the network is split into a number of increments that are loaded into the network in turns. Each increment is loaded into the shortest route, after all the previous increments have been loaded. The link travel times are then recalculated in order to compute which is the fastest route for the next step. In case that more than one route has to be used for the O-D trip, the increments are automatically assigned alternatively to each route. The route becomes faster again after previous increments have been oriented to another route.

Van Aerde and Rakha [5] show that the SO traffic assignment can be solved by considering an incremental traffic assignment. In this paper they study the impact on the traffic network when different percentages of drivers benefit of route guidance. A highway, ramps and an arterial road represent the network. Each of these have a speed limit. The simulation results showed that the availability of route guidance systems to drivers in a congested network reduced the total travel time up to 20%. The majority of the benefits could be achieved for the short incidents scenarios if only 20% of the drivers were equipped with routing guidance.

Results when more than 20% of the drivers were guided were not significantly better. These results are explained by the fact that many of these drivers could have been routed on the same road as they would choose themselves. However, in case of longer incidents the incremental benefits of providing route guidance to more than the first 20% of the drivers yield significant reductions in travel time. This can be explained by the fact that the incident caused a disturbance in the equilibrium which requires the re-routing of more than 20% of the drivers.

2.1.4 Equilibrium methods

Equilibrium methods are algorithmic approaches which assume equal travel times. They are optimal assignments since they are formulated on the basis of linear or nonlinear mathematical programming [15]. The user optimum equilibrium can be found by solving the following nonlinear programming problem:

$$min \sum_{a} \int_{0}^{v_a} S_a(x)dx, \tag{2.1}$$

with the following constraints:

$$v_a = \sum_{i} \sum_{j} \sum_{r} \alpha_{ij}^{ar} x_{ij}^{r}. \tag{2.2}$$

$$v_a \geqslant 0, x_{ij}^{r} \geqslant 0. \tag{2.3}$$

In Equation 2.2 v_a represents the volume of traffic on time unit, $\alpha_{ij}^{ar} = 1$ if link a is on the path r from i to j and x_{ij}^{r} is the number of vehicles on link a. In Equation 2.1 $S_a(x)$ is the average travel time on link a.

E. C. Matsoukis [17] classified the equilibrium assignment methods into: assignments with fixed demands, assignments with elastic demands and combined models.

In this study we aim to focus on the first category and review algorithmic approaches for predicting the resulting flow patterns in a network with an equilibrium assignment and fixed demands. As we already mentioned, equilibrium exists when a driver cannot choose another route to optimize his costs. Thus we wish to determine how traffic between the pairs of zones of a city would be distributed at equilibrium.

The equilibrium assignment algorithm is a weighted combination of a sequence of all-or-nothing assignments (i.e in each step minimum path trees are traced from each origin to all destinations). All trips are assigned to these minimum path trees until convergence is achieved. But this produces a NLP(non-linear programming) problem which is hard to solve, in compare to simulation models for example. Moreover, the approach seems useless for realistically sized equilibrium traffic assignment problems. But this problem can be reduced to a much simpler linear approximation and solved using the Frank-Wolfe [18] algorithm where are less requirements. But this iterative linearization procedure still involves longer computational times than the iterative procedure.

Leblank et al. [19] developed an iterative procedure to solve one-dimensional searches and LP problems whose solution proved to converge each time and close in on the equilibrium flows without excessive computational requirements. This model was tested on a network of 76 arcs and 24 nodes. However, a serious drawback of the models mentioned so far is that they do not consider explicit capacity constraints imposed on individual links and this is quite unrealistic from a transportation engineering point of view.

2.2 Dynamic Traffic Assignment (DTA)

When a time dimension is added at the models previously described then the DTA is obtained. Thus, by including temporal dimensions we can represent the real life traffic situation and compute the traveling time. Literature surveys in this field generally mention two main approaches for DTA: the analytical-based models and the simulations.

The first approach which is the analytical-based approach model considers two time indices: the time at which the path flow leaves its origin and the time at which it is observed on a link. In other words, the approach assumes that the whole time is divided in intervals. Then, *static mathematical analytical control models* are applied to each interval, on the assumption that one interval is long enough so that drivers can complete the trip within that certain time interval. These models are heuristic approaches attempting to solve the mathematical formations of an ideal model that would optimize the travel time for the whole network. But this approach is mostly purely

theoretical and aims at controlling the whole network in order to optimize the total travel time. But to what extend do these models realistically capture all complexities? And, to what extend can more complex models be resolved? Because if more aspects are included then more variables are needed. Moreover, in practice, drivers so far cannot attain such an equilibrium state in the network.

Literature within this area of research is extensive. DTA has evolved a lot since the work of Merchant and Nemhauser [20] who considered a discrete time model for dynamic traffic assignment with a single destination. The model they assumed was nonlinear and nonconvex.

Meantime, researchers became aware that DTA theory was still undeveloped and necessitated new approaches to account for the challenges from the application domain. DTA comes across a large set of problems depending on various decision variables, possessing varying data requirements and capabilities of control.

The second approach is the simulation-based model. This approach simulates the behavior of the drivers in different traffic settings. Due to their capability of better representing the real world they increased their popularity. Simulations usually try to replicate the complex dynamics of the traffic. Although that this is considered a different approach, the mathematical abstraction of the problem is a typical analytical formulation.

Section 2.2.1 describes the analytical based approaches whereas Section 2.2.3 presents some famous simulation environments in literature.

2.2.1 Analytical based approach - Mathematical Programming Models

This section aims to exemplify some analytical-based approaches and mathematical programming models for DTA from literature. These models were studied because of their complexity and manner of expressing the traffic flow in order to achieve a DTA. More insight was gained with regard to the variables and models that can describe DTA. One model is presented more in detail in order to emphasize the basics of this approach.

Mathematical programming models of time-varying traffic flows consist basically of three parts:

- Conservation of the flow;
- Route choice criteria (usually UE or SO);
- A model of flow behavior/propagation - which is crucial for determining the temporal dynamics.

Peeta and Ziliaskopoulus [21] split the analytical models from literature in four broad methodological groups where the first ones are the mathematical programming formulations. Within this approach flow equations are deducted and a nonlinear mathematical programming problem has to be solved. Merchant and Nemhauser [20] and Ho [22] studied such models. Due to the complexity of a nonlinear problem, a linear version of the model with additional constraints can be created and solved for a global optimum using a simplex algorithm. The linear program has a staircase structure and can be solved by decomposition techniques.

Birge and Ho [23] presented an approach to find optimal flows in a dynamic network with random inputs into the system and congestions on the flows. The settings of the model were deterministic. Their purpose was to build a globally optimum solution on the deterministic results, in the stochastic problem, by a sequence of linear optimizations. A decomposition algorithm is presented for this purpose. The model is based on classic traffic assignment methods. An assumption was the control of the whole network in order to achieve the SO. Their algorithm was rather computational demanding although using small networks (thousands of constrains and much more variables).

Seven years later, Ziliaskopoulus [21] uses the cell transmission model in order to formulate single destination optimum solution DTA as a LP. However, the model is limited to one destination and although it takes into account traffic realities (captured by the transmission model) it is not

feasible for actual applications. The purpose of the article was to show that DTA can be modeled as a LP, a fact that opened new horizons.

At the same time, Carey and Subrahmanian [24] try to model link flows more flexible, so that the travel time for a vehicle is determined by:

- The flow rate when the vehicle enters the link;
- The flow rate when it exits the link;
- The speed of the traffic ahead.

In this article the flow links are determined depending on time. The model relies on speed and density on the links and congestion is considered only in terms of flow-dependent travel times. A mathematical model for time varying flows should ensure the principle of FIFO (first in, first out). The model of Carey and Subrahmanian [24] satisfies the principle in some simple cases but if there is a sharp rise in the flow followed by a sharp fall then FIFO is violated. This can be explained also in a different way: if fast traffic is following slow traffic and there is a gap ahead of the slow traffic and the speed difference is sufficiently large then the present model may allow the fast traffic to "jump" over the slow traffic into the gap ahead it. There would be a few solutions to these violations (namely by ensuring that there are no time-space links in the network that cross each other) but the authors chose to ignore them if they are not in critical parts of the network.

In the following subsection more details are going to be presented about this model. First, a single-destination model is build up, taking into account the necessary variables and the constraints for the LP.

A single destination model

The network is represented by a graph $G = \{N, A\}$ where A is the set of directed links and N the set of nodes. The time horizon is divided in fixed parts and for each spatial link we have time-expanded links $(jt\zeta), \zeta = 1, ...T, t = 1, ...T$.

The flow that enters link j in time period t and exists from it in period τ is $x_{jt\tau}$. The number of vehicles on a spacial link at a certain time gives the traffic volume. And this is equal to the sum of the flows that entered before t and have not yet exited at t. The inflow at node k in period t is E_{kt}.

Flow conservation over space and time

Conservation of flow is one constraint and requires that for each node k at time t the total inflow should equal the outflow. Thus, the flow conservation states that,

$$\sum_{j \in in(k)} \sum_{\tau=1}^{t} x_{jt\tau} + E_{kt} = \sum_{j \in out(k)} \sum_{\tau=1}^{t} x_{jt\tau}, k \in N, t = 1, ...T. \tag{2.4}$$

Travel time/ Cost

The total travel time for all traffic entering the network withing the time span of the model is the following:

$$z = \sum_{j \in A} \sum_{t=1}^{T} \sum_{t=1}^{T} x_{jt\tau}(\tau - t). \tag{2.5}$$

It is considered that $s = f(x)$ represents the travel time function. It is assumed that this is a linear function in time since that allows placing breakpoints of $f(x)$ on the links of the time-expanded network. The capacity of a link is $\overline{x_i} = f^{-1}(i)$.

An LP formulation of the system optimum DTA

A program can be written to minimize all the travel costs (Equation 2.5) by taking into account the conservation equation 2.4 and the linear trip time functions:

$$P_a : minimize(z\{a_{jt\tau}\} = \sum\sum\sum a_{jt\tau}\bar{x}_{jt\tau}(\tau - T)), \tag{2.6}$$

where $x_{t\tau} = a_{t\tau}\bar{x_{t\tau}}$ can be interpreted as the flow on the time space link $(t\tau)$. The main issue is to minimize the total travel time:

$$c(x) = (link - flow) \times (link - trip - time) = xf(x). \tag{2.7}$$

Inconveniences

The SO solution of this model tends to delay some users on earlier links by forcing them to arrive at later links when congestion is lower. But the model can be extended to multiple destinations and to solutions that converge to UE. Although the number of variables increases only linearly in this model, for bigger networks the size of the LP P_x, $P_a$and M (which we consider for the total number of variables), can quickly run into tens of thousands of variables and constraints. Moreover, the model is focused on congestions that produce delays whereas queues at the start or at the end of the exits should be considered as well. A possibility to solve this would be that each link is divided into two parts: a travel and a queuing link.

As a conclusion, the substantial research in mathematical programming based DTA approaches usually meets its limitations when trying to develop models for general networks.

2.2.2 Optimal control formulations

In optimal control theory the routes are assumed to be known functions of time and the link flows are considered continuous functions of time. The constraints are similar to the ones at the mathematical programming formulation, but defined in *continuous-time setting*. This results in a continuous control formulation and not in a discrete-time mathematical program.

Friesz et al. [25] discuss two continuous link-based time formulations of the DTA for both the SO and UE objectives considering the single destination case. The model assumes that the adjustments of the system from one state to another may occur while the network conditions are changing. The routing is done based on the current condition of the network but it is continuously modified as conditions change. The SO model is a temporal extension of the static SO model and proves that at the optimal solution the costs for the O-D used paths are identical to the ones on the unused paths. They established as well a dynamic generalization of Beckmann's equivalent optimization problem.

Although this approach seems successful for describing dynamic systems the optimal control formulations for DTA suffer of limitations such as the lack of consistent constraints to ensure the FIFO principle or to avoid holding the cars at nodes. Even more, it lacks a solution model for general networks.

2.2.3 Simulation-based approach models

Simulation environments address key issues of the traffic assignment, such as the flow's propagation in time and the spatio-temporal interactions. Contemporary DTA models were developed using different simulators (such as CONTRAM (CONtinous TRaffic Assignment Model), DYNASMART or SATURN etc.).

SATURN [26] is an early DTA simulation tool that uses an equilibrium technique. It takes into account delays from the vehicles on the same road, the shape of the vehicle platoons, the effect of traffic signals and the capacity of the lanes. The model treats rather platoons of cars than individual cars and it mostly studies the delays at intersections. It consists out of two parts: a

simulation component and a traffic assignment one. This model has been used a lot in studies but it is less suitable for simulations of highways as it focuses a lot on the the delays at intersections.

CONTRAM [27] also has a simulation and an assignment component. It is an approach that treats packets of cars as a single car to which a minimum path is assigned. It takes as input the network and the route demands between a set of origin and destination zones and it outputs the resulting network flows, routes and travel times. Traffic conditions are simulated by dividing the simulation period in intervals. The packets of cars are routed based on the travel time of the previous ones. This is considered an advantage of this model, because the effect of packets leaving later on packets leaving earlier are considered.

The CONTRAM simulation environment is more dynamic than the previous ones as it allows the re-routing of cars if traffic conditions worsen. However, it does not consider a maximum storage capacity for roads and it assigns cars only based on the Wardropian principle.

DYNASMART is a contemporary DTA model which uses the basic CONTRAM concept. Abdelfatah and Mahmasanni [28] show an example of a DTA model developed by the DYNASMART approach. It is a heuristic simulation-based procedure to obtain time-varying assignment of O-D demands. They present a numerical experiment for a simulation-based algorithm. A realistic network for a part of a region and a hypothetical network are used to show the potential of an additional improvement by the joint consideration of signal control and route assignment compared to either one of these decisions alone. Regarding the signal optimization a local area is defined for each intersection that comprises other neighborhood intersections which affect/are affected by the delay at the intersection of interest. Results pointed that the joint optimization can lead to a better network performance but the improvement offered by signal optimization decreases with higher congestion levels. Moreover, the application reflected that the algorithm was likely to find some local optimum which could thus offer better performance than the base solution.

A limitation of the simulation-based DTA assignment is the inability to derive the associated mathematical properties. Traffic simulation models are also rather computational expensive. The use of a simulator as part of a project to predict the future may also be operationally restrictive. This is why many recent simulation models treat the accuracy of the solution with the computational efficiency.

2.3 Dynamic travel time estimation

DTA requires the evaluation of the traveling time for each road in the network in order to assign minimum paths to the requests. So far, there are few commercially distributed systems which provide an approximation of the traveling time based on current traffic information. The information they use derives from the measurements achieved along the highways for the traffic flow. These spot-speed measurements enable the estimation of the traveling time based on the average speed. But the prediction in this case is done based only on the present situation.

Various techniques that attempted to estimate the traveling time were used in research. This problem has been a great challenge because of the multitude of factors that influence the traffic flow, especially the future traffic flow. Factors that influence the estimation of the travel time are presented in Section 2.3.1. Various measurement techniques along with possible approaches to compute the traveling time by using each of them are described in Section 2.3.2. Section 2.3.3 presents some approaches to forecast the travel time which use historical data, current data, time series of traffic variables etc. along with their improvements or impediments.

2.3.1 Factors that influence the estimation of the travel time

Vehicles, drivers and road conditions are the main components of the traffic environment. All factors that influence these elements affect the traveling time as well. Diversified drivers and road conditions could cause large differences in the travel times as well. Even in the same time interval and on the same road different drivers end up with different travel times. Drivers may have their own traveling speed. For example, most people do not want to exceed the limit but there are

others who travel with a lower speed because it is cheaper. This free flow speed affects the travel time.

Lum at al. [29] showed that the average speed depends on the road's geometry, on the traffic flow characteristics and on the traffic signal coordination. A new travel time-density model was formulated by incorporating the *minimum-delay per intersection* and the *frequency of intersections* as parameters. The traveling time and the traffic volume are two main field items that have to be considered for the speed-flow study along arterial roads. The traveling time can be observed in two ways [29]:

- The *stationary-observer* method which implies that fixed observers are placed at fixed locations along the way;
- The *moving observer* method which includes the floating car technique and the chasing car technique.

Most influencing factors that have been cited in literature are the special incidents and holidays, signal delays, weather conditions and the level of congestion.

The prediction error might be also directly proportional with the length of the forecasting period [30]. In this article the authors did some data collection and analysis. Then, they used this data for modular neural networks (MNN). The purpose was to forecast multiple periods traffic features such as speed, occupancy and volume based on previous traffic information. The next step was to determine the expected travel times based on the predicted values. The analysis of the data showed the following:

- Knowing what happened in the near past could be useful to predict what will happen in the near future for a given place;
- Knowing what has happened upstream and downstream of a given detector location could be useful for predicting near term future conditions.

2.3.2 Measurements and analysis of the travel time

Travel time can be measured as aforementioned, in two modalities: by using *site-based measurements* (car plates registration, AVI tracking, remote tracking) or by using *moving observation platforms* (a floating car, volunteer driver or a probe vehicle). More details along with some examples on these modalities and the processings of the data which is obtained are presented in the remaining of this section.

Car plates registration

This method assumes the registration of the cars plates and the arrival time. The travel time is computed by matching the characters of the vehicles plates.

Coifman and Cassidy [31] present an algorithm that matches individual vehicles measured at a freeway detector with previous measurements taken at another one. A problem is that detectors do not measure the vehicles length as well. Moreover, there can be also plenty of errors in the measurements. Because of the drivers tendency of driving in platoons, the algorithm tries to match the platoons of cars rather than individual ones. When an individual car is matched then its traveling time on the interval between the two detectors is measured.

Alternatively, approaches that match cars lengths and restrict to small platoons of cars (2, 3 vehicles) have been developed [32, 33].

Automatic Vehicle Identification (AVI) tracking

Dion and Rakha [34] use a low-pass adaptive filtering algorithm designed for real time estimation of the mean travel time. They use automatic vehicle identification (AVI) recorded data. The algorithm has two important particularities: it can handle both stable and unstable conditions

and it is suitable for both signalized arterial roads and highways. Their approach is evaluated on observed freeway travel times from San Antonio. The system has much lower sampling rates compared to previous filtering algorithms. This is why it can better track sudden changes in traffic conditions. The average travel times are determined for each read from the AVI data. Duplicates and invalid observations are ignored. The expected variance in the average travel time between samples and the number of consecutive data points that are above or below the validity range are used in the process. Moreover, the variability in the travel times along an interval is also used. The approach was successful when using two different datasets. But further investigations should be done, by using historical data as an additional validation criterion in order to validate the resulted estimations.

Remote and indirect tracking

This method uses the vantage points to observe vehicles movements. *Mobile phones* can be included in this category. They are a really promising technique for computing the travel time as long as studies showed that more than 80% of the drivers have a mobile phone.

The *floating car* is also a popular technique which consists out of specially dispatched vehicles to drive with the traffic streams in order to collect data. Probe vehicles can be indeed an efficient way to estimate the current traveling time based on their speed profile.

2.3.3 Methodologies of forecasting the travel time

Most of the short-term forecasting methods that were used in literature can be divided in two categories, namely *regression methods* and *time series estimation methods*. A third category can be described as combining these two. Relevant forecasting techniques examples which belong to previous research studies are presented in the following paragraphs. The type of traffic data that was used along with possible inconveniences that we detected are included.

Historical Data

Hobeika and Kim [35] constructed three models for short-term traffic prediction by combining the *current traffic*, the *average historical data* and the *upstream traffic*. Thy used the following combinations: historical average data and upstream traffic, a combination of current traffic and upstream traffic and a combination of all three variables. Current information is important to infer the traffic trend. The historical average represents the time of the day-pattern and is used in order to smooth the sharp changes in the current traffic flow and thus to avoid extreme forecasts. The upstream traffic is introduced in order to reflect the dynamic nature of the traffic flow. The quality of the prediction model is actually dependent on the way these variables are combined. The models were evaluated with a regression analysis. They were applied to 15 minutes freeway data obtained by regular ILPs (Inductive Loop Detectors). The prediction models did forecast more accurate traffic data for intervals like 15 minutes than for intervals of 30-45 minutes. Furthermore, the model performed better for congested traffic whereas it showed no improvement for non-peak hours traffic. In this case, historical averages used for travel time predictions would have the same performance. Besides that, they use plenty of non-optimal heuristics in order to reach a prediction model. Another assumption that was used can also be debatable, namely if the current travel time along the network is 25% longer than the monthly average travel time then the second model has to be used and otherwise, the third model.

Li and McDonald [36] use GPS equipped probe vehicles and determine mean speed values in order to develop a *fuzzy mathematical travel time estimation model*. The method associates a driving pattern with the difference between the travel time of the probe vehicles and the average travel time from historical data. Travel time is computed by using the corresponding equations for different driving patterns. The driving patterns are derived from the speed profile of the vehicle and classified by the difference between travel time of probe vehicles and the mean travel time as

Table 2.1: Mean Speed performance

Mean speed types	J_{cal}	%	E_{TTS}	%
space mean speed	42.5	100	6.0	100
time mean speed	38.9	91.5	5.5	91.4
estimated space speed mean	42.0	98.8	5.6	93.1
geometric mean speed	39.4	92.7	5.1	85.4
estimated space speed mean '	38.3	90.1	5.5	91.0
estimated space speed mean "	42.3	99.5	6.5	107.9

fast, medium or slow. However, the model is inaccurate for highly congested roads and the authors claim that a greater number of probe vehicles would improve it.

Burger et al. [37] study the effects of using different approximations of the space mean speed in order to use it with a control predictive model. Six mean speed types are discussed, together with the time mean speed and geometric mean speed. The suitability of these mean speeds is tested with a microscopic simulation. Next, the three most suitable mean speeds are used with the predictive model and the time mean speed leads to the best result regarding the total spent time.

In Table 2.1 [37] the mean values J_{cal} and E_{TTS} are calculated for five data sets. The first column shows the average calibration errors, the second column shows the calibration errors as a percentage of the referred value (a lower percentage represents a better fit of the model to the measured data). The third column represents the average error between the measured and predicted TTS (total spent time). The forth column shows the relative error based on the reference value, where a lower percentage means a better prediction of the total TTS.

The three best results are given by the geometric mean speed, estimated space mean speed and the time mean speed.

When tested on the A12 freeway in The Netherlands the use of the control predictive model showed improvements up to 15% compared to an uncontrolled situation [37]. Reducing the shock waves also slightly improved the traveling time. Thus, this article infers a model that controls the speed of the vehicles. The tests on the highway showed a slight improvement.

Time series models

Time series analysis are as well a popular method to infer the travel time prediction due to their strong potential for on-line implementation.

Ishak et al. [38] describe a *short-term prediction model* for speed that follows a nonlinear time series approach and uses a single variable. The variable represents the most recent speed profile at each loop detector station. The model is described as follows: given a time series of $V_1 ... V_n$ it is required to predict the value for V_{n+1} by using data that was collected from point 1 to n. The model has the form:

$$V_{n+1} = (3^{\alpha_n} - 1)V_n - V_{n-1}. \tag{2.8}$$

Depending on the value of αthere are 3 cases distinguished:

- when $\alpha = 1$, there is a stable situation and the future speed will be similar to the previous two;
- when $\alpha < 1$, then the forecasted speed will be calculated as a fraction from the current speed added to the difference between the current and previous speed;
- when $\alpha > 1$, the opposite holds.

It is not necessary that the value of α is always connected to the type of data (stable, concave or convex). The factor that proved to influence the short-term predictions the most was the congestion index (Figure 2.2 [38]).

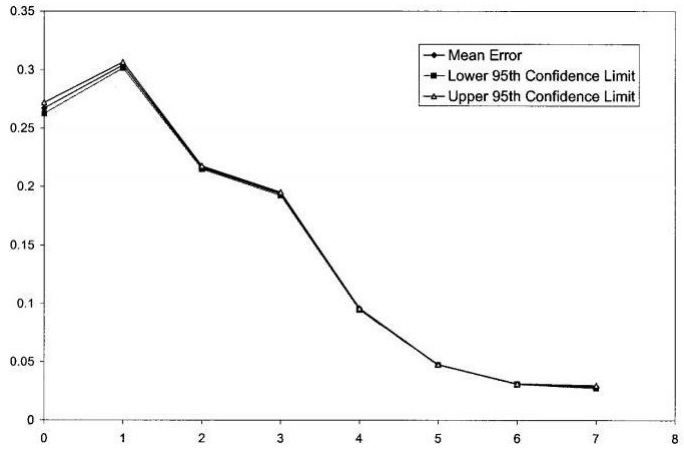


Figure 2.2: Effect of the congestion index on the prediction error

Artificial Intelligence

Artificial Intelligence (AI) gained a great attention also in the traffic assignment. One popular technique in AI applied also for the estimation of the traveling time are Neural Networks. Artificial Neural Networks (ANN) have been applied in a lot of fields and thus for traffic issues as well [30]. Researchers used ANNs with different variables such as *speed* and *volume* in order to predict travel time. Due to the fact that we benefit of current traffic information and we should predict a future situation of the traffic the ANNs can be a promising approach.

In review of literature, researchers have used *parametric models* in order to forecast the travel time, such as *regression models* or *time series* and *nonparametric models* that include *ANN models* [39, 40, 41]. Studies have shown that ANNs (including modular neural network model and state-space neural network model) are a powerful tool to predict travel time on freeways [41, 40].

Yu et. al. [39] proposed a *travel time prediction model* which comprised two parts: a *base travel time* and a *travel time variation*. The first term is computed by using a fuzzy membership value average of the clustered historical data that reflects the traffic pattern. The variation is predicted through a cluster-based ANN in order to capture the traffic fluctuation. The model was tested using simulated data. The approach proved to be reliable. However, the model should be also tested with field data under more complex traffic conditions.

An important fact would be that a mixed-structure neural network is able to predict the travel time for segments of roads where there are no detectors and this is based on the data from segments with detectors.

2.4 Incident effects on traffic congestion

The prediction of delays in case of incidents is a challenging problem to solve [42]. Most of the researchers developed macroscopic dynamic models in order to estimate the basic variables in traffic, such as the traffic flow, the speed and the density. Still, abstract models of the real life situation are used and most of the approaches do not take into account the effect of queuing and lane changing.

Section 2.4.1 briefly emphasizes the important delays prediction variables while Section 2.4.2 gives more insight into statistical models used to predict incident delays.

2.4.1 Incident delays prediction variables

There are two significant variables related to incidents, namely the prediction of the delays and the length of the queues.

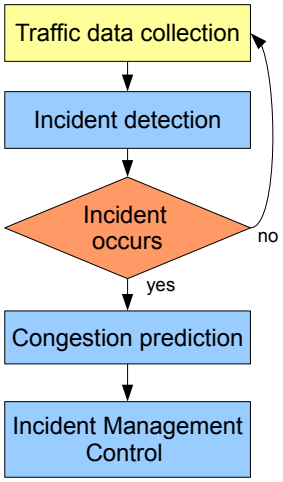


Figure 2.3: Framework for incident management

Sheu et al. [42] infer a stochastic model for the estimation of these two and for lane-changing fractions. In order to test their approach they used the data from a simulator (Corsim) where they simulated different types of incidents.

Figure 2.3 [42] roughly presents the layout for an incident management framework. This involves three main parts:

- Incident detection;
- Delays prediction;
- Prediction of the congestion in time;
- The incident-responsive traffic management and control.

The changing of lanes at incidents and their combination with the acceleration and velocity decrease process delay the vehicles [42]. The variables supposed to characterize the congestion in real time in case of an incident are the following:

- Lane traffic state variables (which can be estimated on raw data collected);
- Number of moving vehicles on the road;
- Queues length;
- Delays.

Different types of equations compose the discrete-time nonlinear stochastic traffic model, such as: recursive, measurement equations, delay equations and boundary constraints. The differences between the mean of the estimated and simulated values proved to be acceptable according to the authors [42]. The estimation error with regard to the lane-changing fractions was also studied. The results showed that there is potential in the approach but further studies have to be done.

2.4.2 Statistical models used to predict incident delays

Incident delay models proposed by Garib [43] showed that 74% of the variation in incidents delay is determined by: the duration of the incident, the number of cars that are involved, number of lanes that are occupied and the traffic demand before the accident happened.

Garib et al. [43] proposed two statistical models (regression analysis) to estimate the incident delay and a model to predict the duration of the incident. The duration prediction model showed

that 85% of the variation in the incident duration is determined by how many lanes are stuck, whether there is a truck involved, the time of the day, the response of the police and the weather conditions.

In the article of Hounsell and Ishtiaq [44] an *incident database* was compiled from a simulator (CONTRAM) on a range of networks, traffic and incident scenarios. The approach considers statistical models for predicting the number of affected links including the one with the incident. Another inferred issue is the change of the traveling time on the incident's link and on the affected links. An algorithm was constructed that computed a backward tree in order to obtain the affected links. Models were developed that would forecast the journey time in the new conditions, after being supplied with the severity of the incident (meaning the reduction in the capacity of the road) and the location in the network.

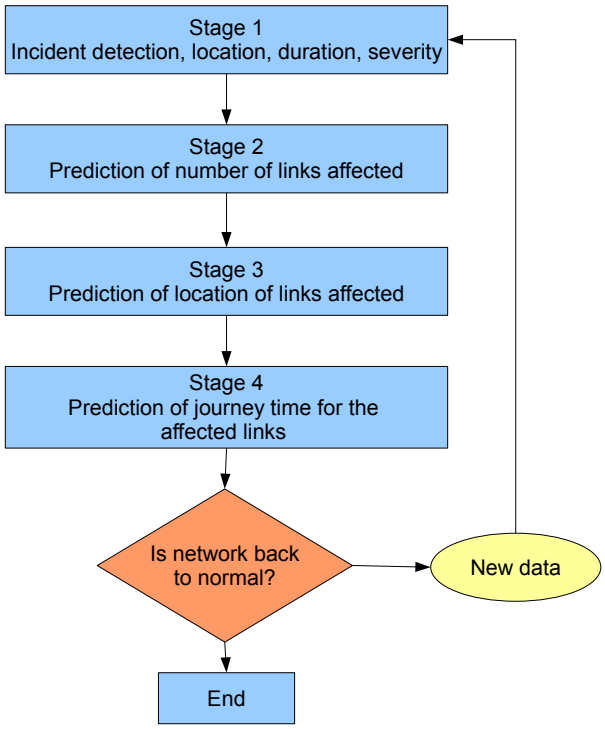


Figure 2.4: Flow chart of incident modeling procedure

The simulation database

Two networks of 150 and 190 links with 55 and respectively 71 junctions were used for simulating various scenarios [44]. Two links from each network were used to place the incidents. The scenarios depended on the location, the incident severity and on the incident's duration. For the compilation of the database it was assumed that drivers have fixed routes, without being capable of diverging from the routes.

Traffic modeling in the neighborhood of the incident

A flow chart of the main stages for incident modeling is presented in Figure 2.4 [44]. In the first stage, the incident is located together with the severity and duration. In the second one, the

number of affected links is predicted. The third step involves finding the affected links. Finally, the new travel times for these links are predicted.

The severity of the incident proves to be a dominant parameter with regard to the number of affected links, by using multivariate statistical analysis. In order to detect the location of these links in the network the nearest upstream links to the incident link are first considered. Then, the propagation continues in the same manner until the maximum number of links is reached. When the incident ends the number of affected links will decrease following the reversed process.

The obtained results were compared to the data generated by the simulator [44]. The first issue was the lack of update. The effects of the incident were predicted at the start of it for the rest of the affected periods. For real time applications on-street information would be necessary.

2.5 Summary

After we presented a classification of methods approached in literature for traffic assignment it is time to get back to the main problems that we want to discuss, presented in the introduction chapter of the thesis. We studied various proposed models and frameworks in order to gain more insight into the main factors that influence the traffic assignment. While doing this, we highlighted the various scientific challenges of the development of a traffic model. The advantages and the disadvantages of various methods were described and analyzed.

As we can notice already in the first section of this chapter, a complex problem is to satisfy the users preferences while using the capacity of the traffic network efficiently. This means that usually there is a trade-off between the two. We discussed many equilibrium and non-equilibrium methods and showed, from examples, that a User Equilibrium solution does not include a System Optimum. So, a big challenge is how to develop a model which would satisfy both of them. But given the proves that the first one does not necessarily include also the second one, which remains the optimum choice?

Researchers added a third dimension to the traffic models and obtained the dynamic assignment. Most popular methods that we discussed in this chapter are either analytical formulations or simulation based models. The mathematical programming models usually proved to be computational demanding due to their complexity. That is why there is always a trade-off between accuracy and tractability.

Furthermore, we were interested in the research regarding the estimation of the traveling time. We read articles/journals that focused on future predictions. The most successful methods estimated the traveling time by using historical data as baseline and real time information. But results were improving when adding a potential prediction of the upstream traffic. There are various approaches in order to asses the latter one, such as ANNs, statistical methods etc.

Moreover, we studied the factors that influence traffic congestions besides the recurring ones at peak hours. The main factor is represented by incidents. But the weather conditions, the day of the week, the time of the year or special events can influence traffic congestions as well. With regard to incidents management we discussed some statistical approaches and presented the most relevant variables of an incident management model.

Some simulation based models were also described because simulators are important in order to better understand the traffic and measures that can improve the traveling time. They are also used to test and evaluate prototypes or in order to generate data.

State of the art research showed that nowadays a lot of effort is invested in developing intelligent highway systems that would satisfy the needs of the future society. Such a generalized system that handles the vehicles and combines this with roads sensors and dynamic routing is still beyond the possibilities that we have nowadays.

However, as we can experience ourselves, traffic management still needs to be improved. And this may demand a centralized routing service that supervises the vehicles which are in the network.

Chapter 3

Industrial products

Studying the state of the art industrial products represented a necessary step in our research. Due to the fact that we aimed to design our own personal advanced traveler assistant that would ideally be used by a large group of people, we were interested in the latest systems that have similar purposes/intentions.

We start the chapter with PITA, in Section 3.1, which is an academic concept developed at Delt University of Technology. As we partially based our work on the research results of PITA we wanted to present their ideas. We continue with intelligent vehicles frameworks which are an ambitious approach to solve the traffic issues. However, most of their assumptions are far beyond the possibilities which are available nowadays. The description of different projects in this area can be read in Section 3.2. Section 3.3 presents the "Mobile Century Project" recently developed by Nokia and Berkeley University, which uses cell phones as traffic sensors. Moving on, the latest feature released by TomTom is presented in Section 3.4.

Traffic management schemes used in The Netherlands for the last 10 years are presented in Section 3.5. We emphasize the most interesting aspects related to each measure.

3.1 PITA

Within the SMM (**S**eamless **M**ultimodal **M**obility) research program of TRAIL (Netherlands Research School for **TRA**nsport, **I**nfrastructure and **L**ogistics) at Delft University of Technology research was conducted for PITA (**P**ersonal **I**ntelligent **T**raveler **A**ssistant) project [45]. This project aims to develop new public transport services for the 21st century.

PITA represents an integrated system that links information services in order to plan a trip schedule by using different transport modalities, inform the user of unexpected delays and offer alternative route solutions.

PITA assumes the existence of a handheld device that provides ubiquitous communication between users and services like the Dutch railways, other public transport and routing services.

Research into PITA focused on the development of a multi modal dialog manager between users and information systems. More precisely, on how multi modal interaction can be modeled in a single dialog manager. This is presented in Figure 3.2 [45].

PITA is a person-linked mobile travel information service that gives the traveler updated information on travel options, also during the journey itself. The environment of PITA is depicted in Figure 3.1 [45]. The services would be developed on the concept of chain mobility where a chain manager would be responsible for the connection between different transports. The traveler would be door to door transported without having to worry about delays or transport connections. One important aspect about PITA is that it relieves the user from finding the best route to his destination. After determining the position of the user by GPS or other techniques the system is able to give an optimal route and update the advice as the user travels and new information about delays or congestions becomes available.

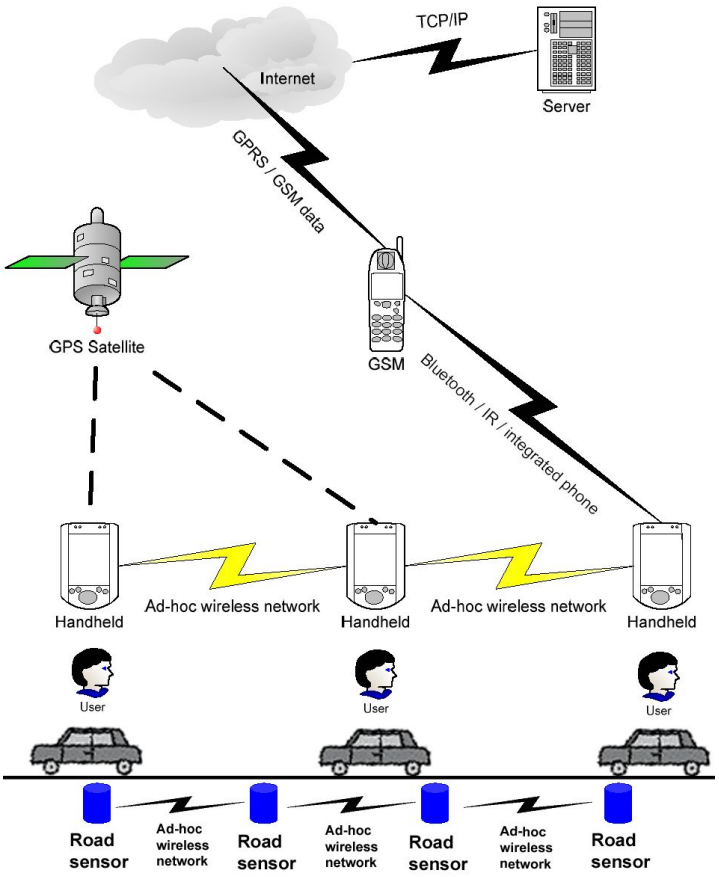


Figure 3.1: Different communication networks involved in PITA

PITA should provide the user with a multi modal interface. This means that the communication can be done in more ways such as by text, speech, by using icons or a graphical interface. In order to optimally fit to the user's desires the system keeps track of several preferences of the user.

In order to offer a synthesized description of PITA's properties we present the following list:

- *Traveling Assistant.* PITA would continuously update the user with the best route to his destination depending on his current location, on the available delays information and his preferences.

- *Intelligent Assistant.* PITA is capable to determine the delays in the network and their consequences on the user's route. The consequences are determined by longer traveling time or unexpected incidents on the way. The system should generate alternative routes, estimate the new traveling times for them and compute the risks of changing to other routes. Then, it offers the user a solution. Moreover, it is possible to communicate with PITA in a multi modal fashion: by speech, text or graphically. Depending on the situation, PITA or the user can choose the most convenient route.

- *Personal Assistant.* PITA runs on a personal handheld device. It is capable of learning the user preferences and act accordingly. This device is envisioned to be similar to a digital copy of the user's preferences, that manifests in order to compute the best route.

- *Determines the user's location and the delays.* Essentially for PITA is the retrieval of real life delays information. But the possession of such information is not straightforward. There

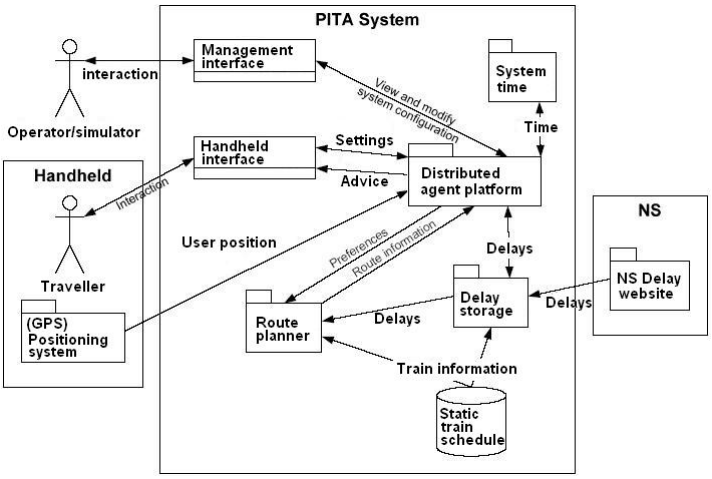


Figure 3.2: PITA system design

are various traffic information sources but PITA should rely on the GPS equipped devices, on the ad-hoc networks among these and the main server.

3.2 Intelligent vehicles (IV) traffic management frameworks

Automated Highway Systems (AHS) and *Intelligent Transportation Systems* (ITS) are state of the art in the field of traffic management. The need of such systems appeared due to reasons like the high number of accidents that occur, injuries and deaths that result from these, congestions and loss of time in queues [46].

Intelligent vehicles (IV) is a concept that sums the features offered by AHS and ITS. IV field includes various control mechanisms such as automatic cruise control, speed adaptation, dynamic route planning and other feature discussed in Section 3.2.1. While discussing IV another concept that appears is the concept of "platoon" , which will be described later in Section 3.2.2.

Developed traffic management architectures such as PATH, Dolphin, etc are discussed and an IV-based traffic control framework that combines the strong points of these is presented. A guiding survey on traffic management and control frameworks that are based on IVs is presented in [47] by L.D. Baskar et al.

Even if the work is promising there are several drawbacks in this approach like introducing more accidents due to failure. A discussion of these drawbacks is presented in Section 3.2.4.

3.2.1 Relevant features of Intelligent Vehicles

The IV applications are divided by Baskar et. al. in three main categories [47]:

1. advisory systems (optic / acoustic) which can warn the driver;
2. semi-autonomous systems that use different measures to take partial control of the vehicle (e.g. the "active accelerator" which interfere when the driver excels the speed limit so that he has to exert more pressure on the pedal in order to attain a higher speed);
3. and fully autonomous systems that take over the control of the vehicle.

With regard to the IV control measures, the main components are the ACC (Cooperative Adaptive Cruise Control), the ISA (Intelligent speed adaptation) and the dynamic route planning. The ACC controls the longitudinal motion of the vehicle and maintains a minimum gap between vehicles. The ISA assures that the driver does not exceed the maximum allowed speed, being capable to figure out which are the restrictions. Dynamic route planning includes traffic conditions such as

traffic jams, dynamic speed limits and frequent updates of the travel times based on real time traffic data.

3.2.2 Platoon-based traffic models

Arranging cars in platoons is a functionality that allows "hands free" operations in traffic [46]. Using the platooning concept, high speeds and short distances among vehicles can be maintained [46, 48].

Broucke and Varaiya [46] presented an *architecture* that would allow an increase of the vehicles capacity on roads, that would reduce the travel time and assure collision-free operations (if there are no malfunctions). They also claim that such an architecture would reduce emissions and thus the pollution. "Platooning" here assumes smooth merging, lane changing and splitting. The architecture is a five level hierarchy containing:

- the *network* which determines the route of the trip from the origin to the destination after estimating the highways state based on sensors;
- the *link layer* which manages the roadway's parameters (e.g. speed, flow control at entrances/exits);
- the *coordinator layer* which does the management of cars (e.g. lane changing);
- the *regulation layer* which executes the cars maneuvers by control laws for lane keeping, lane change etc;
- and the *physical layer* which represents the vehicle dynamics which are controlled by the regulation layer.

The performance of the system is analyzed with regard to the traffic capacity increase (four times the manual capacity), with regard to safety, the coordination level and emissions (which roughly doubled improved).

Kun Li and Petros Ioannou [48] mention that a feature like the *Intelligent Cruise Control* (ICC) allows vehicles to longitudinally follow each other. A traffic model for such cars is relevant in order to assess the effects of vehicle automation and to detect control strategies to improve efficiency. A mesoscopic model in this paper describes the speed and density continuous in time and the microscopic characteristics of the traffic flow. The macroscopic model describes the average speed and density at each point in time for different lane sections. Some simulations are run in order to analyze the effectiveness of these models for traffic flows regarding two scenarios on one lane for an 8 kilometers highway. The speed fluctuations are analyzed. Once the number of cars increased the complexity of both microscopic and mesoscopic models increased significantly and this determined the need of a macroscopic model that considers the average speed and average density generated by the mesoscopic model over sections of the lane. The model obtained in this way is far simpler computationally.

3.2.3 Intelligent Vehicles frameworks

In the following paragraphs we aim to briefly describe some of the most popular examples of IV traffic models.

Dolphin Framework

The Japanese Dolphin framework developed in 2000 [49] is mainly interested in studying the communication between vehicles and in controlling the platoon maneuvers without incorporating intelligence in the road side. It is made out of three layers: the traffic control layer, the vehicle management layer and the vehicle control layer.

PATH framework

The PATH framework proposed by the University of Berkeley in 2005 assumes vehicles are in platoons in order to increase the roadways capacity and safety [46, 48]. It assumes a network of interconnected highways which are divided in links of approximately 5 km and links divided in sections. The architecture's network is similar to the one mentioned in Section 3.2.2.

The Collaborative Driving System

The Collaborative Driving System mainly inspired by the previously mentioned architectures uses ACC technologies to support platoon-based driving. The framework aims to use an inter-vehicle coordination system that ensures the stability among platoons of cars. The architecture contains three layers:

- Traffic control layer which gets the information about the traffic situation.
- Management layer which is responsible for the movements of each vehicle, for the traffic control and for the inter-vehicle communication (which does the coordination and the planning).
- A guidance layer which senses the states of the cars and transmits to the management control the vehicles parameters such as steering, breaking, accelerating etc. .

CarTalk2000

CarTalk2000 [50] is an European project that focuses on driver assistance systems based on inter-vehicle communication. Its main objectives are the development of cooperative driver assistance systems and the development of an ad-hoc radio network as communication basis.

SAFESPOT

SAFESPOT, another European project aims at improving safety on roads using intelligent vehicles and intelligent roads. Its safety allowance can detect dangerous situations in advance and can provide awareness about the surrounding environment. It is based on vehicle to vehicle and vehicle to infrastructure communication.

Transumo

Transumo (TRANsition SUstainable MObility) is a Dutch platform for over 150 companies, governments and knowledge institutes that cooperate in the development of knowledge with regard to sustainable mobility. Using this platform Baskar et. al. [47] propose an improved framework that extends the existing architectures in several directions, with the objective of integrating the intelligence of roadside infrastructure and IV systems with automation. Vehicles are assigned to platoons rather than segments. The main improvements of this framework are the following:

- Vehicles communicate with each other and the infrastructure in an integrated way.
- Both IV-based traffic control measures (such as ACC, ISA etc) and roadside traffic control measures (such as ramp metering, variable speed limits, dynamic route guidance etc) are used in an integrated way.
- The platoon size is one variable that is optimized and it can vary from 1 to a large number depending on the traffic condition and the given traffic performance criterion.
- The framework is integrated with a control model that permits optimal traffic control measures.
- The framework is suited for both inter-urban and intra-urban networks.

In order to deal with larger networks the framework consisted out of several layers with controllers:

Figure 3.3: Test cars - Mobile Century

1. *Higher level controllers* - which refers to regional controllers or controllers for a group of areas.

2. *Roadside controller* - which uses IV-based control measures like ISA for assigning the desired speed to the platoon etc.

3. *Platoon controller* - which is responsible for the coordination inside the platoon, it receives commands from the roadside controllers and it mainly executes the maneuvers inside the platoon.

4. *Vehicle controller* - which is present in each vehicle and receives commands from the platoon controllers. Once the framework was developed an implementation on a small scale setup is mentioned for the future work.

However, although these traffic modeling frameworks are state of the art approaches to solve the traffic issues there are also a few drawbacks, which are presented in the next section.

3.2.4 Drawbacks of Automated Highway Systems and Intelligent Transportation Systems

Although researchers view AHS in a very optimistic light there are serious matters that have to be considered such as the influence of AHS in road safety, which can be compromised due to hardware/software failure. These issues limit the use of such systems in practice. Because safety is a crucial requirement and also because the large investment needed for ITS implementation in a real scenario, society nowadays cannot benefit of these systems. There are lot of precautions to be taken before using such an application on the freeways. As no such system is fully implemented most research is evaluated through simulations.

Even if AHS along with ITS are deemed to decrease the number of accidents what happens if drivers get less alert in time? The chance of committing errors increases after they leave the AHS, and this is a problem as most automated systems are and will be placed on highways or arterial roads and not in all the network. There are many aspects that still remain unknown [46].

The financial issues are also a problem: who would pay for the road side systems? Who would pay for the necessary car upgrades? Is there a real need? Will there be enough demand to cover R&D costs? All these questions make funding and developing these systems a hard task.

3.3 Mobile Century Project

Initiated in 2008 by Nokia in collaboration with the UC Berkeley College of Engineering, Navteq, California Center for Innovative Transportation and The California Department of Transportation

Figure 3.4: Demo of Mobile Century at UC Berkeley

(Caltrans) the Mobile Century project aimed to design and test a state of the art system that collects GPS data from GPS-equipped mobile phones and estimates traffic conditions in real time. This is done in a way that allows the system to preserve the privacy of the users, while still giving enough data to make good estimations.

This was the first demonstration of a real-time permanent monitoring system capable of using GPS data from thousands of vehicles in the traffic to construct velocity fields and travel time estimates. The system was designed to monitor highways and arterials.

One of the first goals was to test this system and to collect data on the field. Therefore 100 vehicles were equipped with special GPS enabled mobile phones (specifically Nokia N73 phone) and were sent out in the traffic. They drove along 10 miles on a highway at different times between 9:30am to 6:30pm (see Figure 3.3). The phones stored vehicle speed and position information every 3 seconds. These measurements were sent wirelessly to a server for real-time processing, which include the estimation of speed along the freeways.

3.4 TomTom

The latest feature developed by TomTom is a *life traffic update*. This is a service that TomTom provides for a monthly cost. In the following paragraphs we will we briefly present how it works and what are the basic ideas.

Traffic updates are transmitted through radio using the RDS-TMC (Radio Data System-Traffic Message Channel) protocol. This is a radio service that delivers real-time traffic information to the navigation device. An RDS-TMC receiver (an accessory available for all TomToms) receives the traffic data, decodes it and translates it into visual or audible traffic alerts. Warning symbols appear on the screen, so that people could see where the congestion is. As the data is directly integrated in the device, an alternative route is offered.

Theoretically, the data related to traffic flows, incidents, weather etc. can be gathered from a variety of sources (traffic monitoring systems, emergency services, motorists calls, etc), then collated at a central information center and passed on to a TMC information service provider. TMC messages contain a considerable amount of information, such as:

- Identification: what is causing the traffic problem and how serious it is;
- Location: the area, the road or a specific location that is affected;
- Direction: the traffic directions affected;
- Extent: how far the problem stretches back in each direction;
- Duration: how long the problem is expected to affect traffic flow;

(a) (b)

Figure 3.5: TomTom with live traffic update

- Diversion advice: alternative routes to avoid the congestion.

The service provider encodes the message and sends it to FM radio broadcasters, who transmit it as an RDS signal within normal FM radio transmissions. The TMC decoder inside the TomTom then decodes the message and presents it as a visual or spoken message.

Some critics state that because of the implementation details of the whole system, accuracy can be low for individual receivers, and even misleading due to the fact that the location sent in a message is only an approximate location.

3.5 Traffic management schemes in The Netherlands

Queue Warning

In The Netherlands travelers are alerted about congestion, queues, incidents, bad road conditions by flashing lights and speed signs activated on variable speed limit signs, as shown in Figure 3.6. This warning system deployed in 1981, with lane control and speed limit signs generally situated every 500 m on dense roads is also referred to as a motorway control and signaling system [2]. Queue warnings intend to reduce the occurrence of secondary incidents and also to provide protection in known zones for bottlenecks. The standard speed limit on highways in The Netherlands is 120 km/h but it can drop to 80 km/h, 70 km/h or to 50 km/h if a sudden shock wave is detected. The speed signs also drop the speed in case of bad driving conditions due to snow, icy roads, heavy rain or fog. The functions of the signalizing system are to provide road services to travelers or to mark closed lanes in case of accidents or work zones. After the installation of the system a 10% drop of the number of accidents on the highways was observed [2].

Ramp Metering

Since 1998, the main functions of this measure are: to alleviate congestion, to lead to a better merging with the traffic stream and to discourage the "rat running" [2]. A "rat" is the name given to a car that speeds on the ramp and wants to get on the highway. This can have adverse effects if the other cars in traffic will not allow it to enter the flow. Assessment studies showed that using ramp metering led to an increased speed on the highways, reduction of shock waves and a decrease of the accidents.

Dynamic route information panels along the highways

The notion of dynamic routing firstly appeared in 1990. In The Netherlands the term Dynamic Route Information Panel (DRIP) stands for electronic information panels. These panels on the highways are intended to provide en-route information on queues, length of queues, major incidents

Figure 3.6: Dynamic speed panels in The Netherlands

and possible, appropriate alternatives. On certain locations, especially in the neighborhood of large cities DRIPs give an estimate of the travel time along the ring of the city. The intention is to influence the route choice and thus improve the traffic flow. DRIPs are supposed to provide the users with a more successful trip. These panels are primarily useful for people with knowledge of the road network. The other ones might get confused if they have to change their route without having further guidance. Anyway, they can anticipate better on queues and obstructions.

(a) (b)

Figure 3.7: Dynamic route information

In Figure 3.7 we have both a graphical and a text DRIP.

However, when it was implemented it was expected that DRIPs will be replaced by personal routing systems in the vehicles. Nevertheless, in the last two decades these panels have been the most reliable source of en-route traffic information for the Dutch drivers.

The Dutch law does not require the government to provide traveler information directly to users. Instead, information is sold or provided to independent information service providers who repackage that information and disseminate it through various sources [1].

We conclude that DRIPs improve the traffic flow and they have met their expectations although they determine a maximum of 6% of the drivers to change indeed their route choice.

Chapter 4

Traffic Data

Day by day the complexity and volume of acquired traffic data is increasing. This is due to the progress that has been achieved in highway based measurements and to the multitude of traffic sensors.

The raw data collected from the highways indicates the average speed of the vehicles at fixed time intervals and the number of vehicles that pass. After an initial processing and analysis it is used either in statistics, for research or for other purposes. More details about historical data can be found in Section 4.1.

In order to measure traveling time there are various modalities, such as AVI tracking, car plates registration or inductive loop detectors. In The Netherlands the information used in traffic management is mostly obtained from inductive loop detectors (ILD) placed in the pavement of the highways. Section 4.2 presents how the traffic data can be collected by ILD. Section 4.3 explains how this data in analyzed at the large centers in The Netherlands. The live traffic information service from ANWB is presented in Section 4.5. Moving on, Section 4.4 presents the RegioLab Delft project where traffic data is acquired and analyzed from the South Holland region.

But the main traffic databases that are formed from the monitoring systems are not open source. That is why, the historical data that we used was extracted from ANWB. We present this process in Section 4.6.1 along with a brief description of the main data structures that we'll use in our database.

4.1 Historical Traffic Data

Historical data may consist of single vehicles trajectories or it may be in the form of databases of traffic variables measurements recorded at spots on the roadways. The broadcasters along the roadways identify and report the traveling speed of vehicles at fixed time intervals, the number of vehicles or the congestion level. But most of the traffic measurements infer the traveling speed which is most important for detecting the traveling time.

Historical traffic data is mostly used for traffic statistics and research as the evaluation of traffic has always been an important subject. A lot of traffic management measures were evaluated by using historical data. These studies result in numerous evaluation reports, describing large measurement programs and thorough statistical analysis. But the initial use of historical traffic data was in research where people used it in order to understand the dynamics of traffic and develop dynamic traffic assignment models. The main purpose was to ensure an efficient use of the roadways network capacity. The traveling time is usually the most important parameter used in research from the historical data. For research in traffic field the traveling time is usually the most important parameter that is studied from historical data. The traveling time depends on the vehicles average speed, on the number of vehicles and lanes of the roadways. In other words, when the speed decreases, the time increases.

The raw data is processed in order to obtain traffic indicators, such as the average speed or the congestion level on the roads at fixed intervals. Missing data is usually computed by interpolation

from the surrounding data (if it does not exceed a certain interval). Other problems might come up at the analysis stage because if the recordings are not done for each lane there are differences between trucks (which have a different speed limit) and cars.

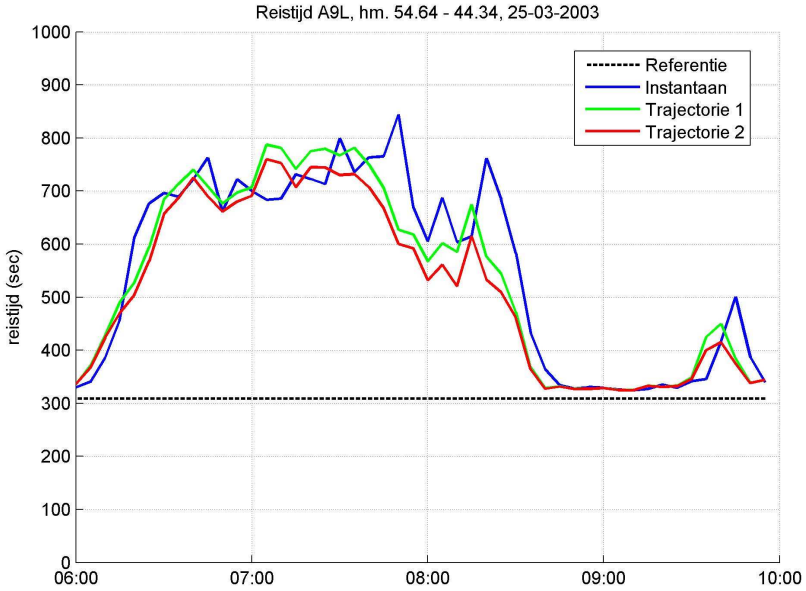


Figure 4.1: Example of travel times in The Netherlands from the MoniCa data

An example of a travel time plot obtained from historical data on highway A9 on the 25th of March 2003 is given in Figure 4.1 [3]. The traveling time expressed in seconds is outlined between 6 and 10 o'clock in the morning. We notice a longer traveling time for the cars starting before 7 and ending after 8 o'clock. Nonetheless, the traveling time seems to increase also before 10 o'clock even if that hour does not represent a rush hour anymore.

The future predictions can be classified into short term and long term predictions. In the long term traffic patterns are recurrent (such as the rush hour in the afternoon). The patterns comprised in historical data make the long term time series based traffic predictions possible. On the other hand, current data represents a basis for short term predictions. For example, by employing information about the current traffic state we can to predict the next couple of minutes. The methods used with this purpose are the fuzzy mathematical travel time estimation models, ANN, Bayesian networks etc. If we consider the attempts in literature to predict the future states on the short term (Section 2.3) we notice that this is still a big challenge. Results are more successful for short interval predictions, such as 10 minutes but for longer intervals it gets more complex. A solution for this would be to know at least a part of the vehicles routes in the network.

The biggest problem is not how to gather current data but how to manage it. Given that nowadays we have huge databases of data, a straightforward question is *how to use historical data efficiently*. Another problem is how to combine all sources of information in order to develop a successful travel time prediction. Moreover, there are many other factors to be considered besides the actual traveling time when trying to predict the future (incidents, bad forecast, events etc). For example, in order to predict traffic for a sunny Monday, a possibility would be to average and use the data for all sunny Mondays from a period in the past. Hence, a challenge is also to detect the best fit of the historical data, depending on the objectives of the research.

The algorithm that we use in our model needs traveling time in order to compute the solution of a route request, based on historical data. Thus, the data that we integrated in our model uses the average speed on the roadways collected in 4 days. The traveling times were assessed using

the average speed for each road and interval of the day. The use of real historical data allowed us to have a realistic experience with the dynamic behavior of traffic. Moreover, we could use it in order to test the algorithms, design an intelligent traveling assistant and implement and test a prototype of the assistant.

4.2 Inductive loop detectors

Traffic data can be collected by a variety of data sensors, such as *inductive loop detectors* (ILD), *videos*, *floating cars, remote traffic microware sensors* etc. The latter represent a relatively new technology for collecting traffic data. But since it is still in the testing stage, only a limited number of such sensors have been installed in the United States. Therefore, it cannot be used for wide-area data collection. Video imaging systems have interested transportation researchers in the past decade, but due to their limitations under unfavorable weather conditions, such as bad lighting conditions, they cannot collect good traffic data continuously. Even under favorable lighting conditions, the video imaging technology has not advanced enough to accurately collect vehicle-classification and speed data. Therefore, the inductive loop detection system remains the main traffic data source.

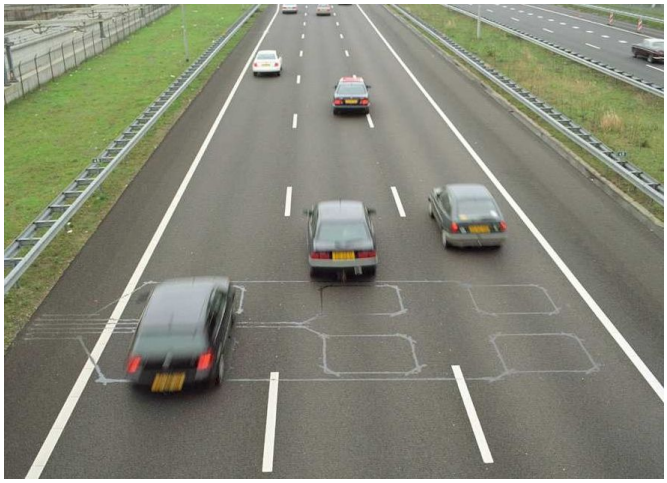


Figure 4.2: Paired loop detectors wired in the pavement

In The Netherlands due to the increasing congestion the traffic management measures proved to be an important direction. That is why since 1980 the monitoring system which includes inductive loop detectors (ILD) succeeded to cover 1000 kilometers of the motorway network [3].

ILDs are the most common technology used nowadays on highways and arterial roadways to collect real time data on vehicles presence, average speed or cars length. Thus, the ILD system is the main traffic data source. As one of the most popular automated traffic data collection methods, ILD technology was first introduced for detection of vehicles in the early 1960s, and today, after a 40-year evolution, it has become a ubiquitous mean for collecting traffic data.

Loop data is generally used to update traffic management measure systems such as queues warning or dynamic route information panels on highways. It is also used to determine the level of congestion or to detect incidents. The ILDs are already used in more than thousand of places along the roadways network in The Netherlands for developing traffic management schemes.

How does an ILD work? An ILD consists of a loop which is several meters wide made of twisted, insulated wire in the pavement. Loops may take a big variety of shapes like circles, diamonds, squares, rectangles and they work by detecting the change of inductance. They detect a vehicle as it passes over the loop by measuring the vehicle's magnetic field.

Figure 4.2 [3] shows an example of paired loops detector used on the highways in The Netherlands for traffic monitoring. Loops are placed at about 500 meters apart on busy roads and less

dense on others. These loops detect the passages of cars and send this to the detection stations that calculate speeds, flows and vehicles classification. The monitoring system gives a continuous view of the actual conditions on the roads network.

ILDs are frequently deployed as single detectors, as one loop per lane, or as speed traps (also called dual-loop detectors or dual loops) formed by two consecutive single-loop detectors placed several meters apart. Single-loop detectors are used to measure volume and lane occupancy, while dual-loop detectors measure speed and vehicle length.

In The Netherlands since 2005 detectors detect the passage of cars and send the information to the detector stations for each lane. Before that, the data was always available per cross-section and sometimes per lane. The measurements are converted into data per minute and sent to the regional traffic control centers. They are typically done for a certain minute and files have special names that indicate the location of the measurement, the date, the speed, the congestion indicator and the flow [3].

As aforementioned, the monitoring system served firstly for queue warning, incident detection or as a tool for supporting the road works in case of closed lanes. Besides this it is also available for other purposes, such as research or statistics. At the control centers the data obtained from monitoring is organized in files and databases.

The traffic data which is collected from the IDLs belongs to the Ministry of Transport and Water Management and it does not represent a public source of data.

4.3 MoniCa Data and MoniGraph

The traffic control centers in The Netherlands collect the traffic data from the detector stations in their region. The country is divided in 5 regions. In the control centers the data in organized in files and databases. Each file contains the data in ASCII format for every measurement location in the region for a certain minute. The name and extension of the file represents the date, time and the region. This is called the *MoniCa data*.

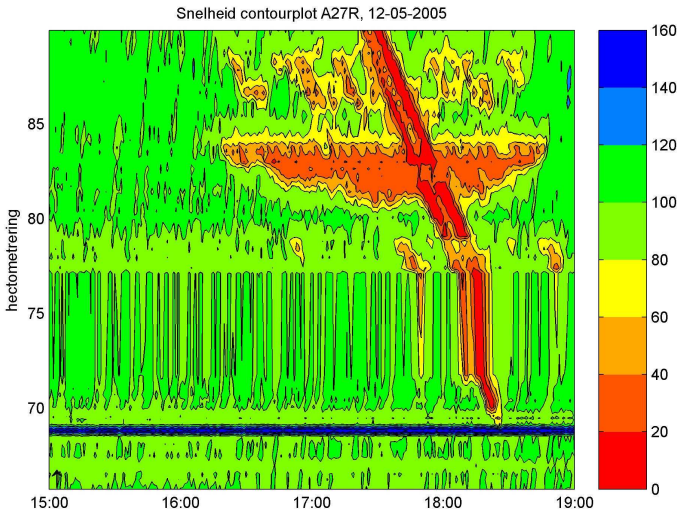


Figure 4.3: Analysis of data - Speed contour plot

The tool used recently by the Dutch for processing the traffic data into traffic indicators is called the *MoniGraph*. This tool processes, visualizes and analyzes the monitoring data. The input consists of the day and time period for which the analyze is done, the location and other analysis parameters. The output consists of graphs on speed, flows and travel times along with

information about the quality of the data and the network indicators. These may be the total distance traveled, total delay, total congestion, etc.

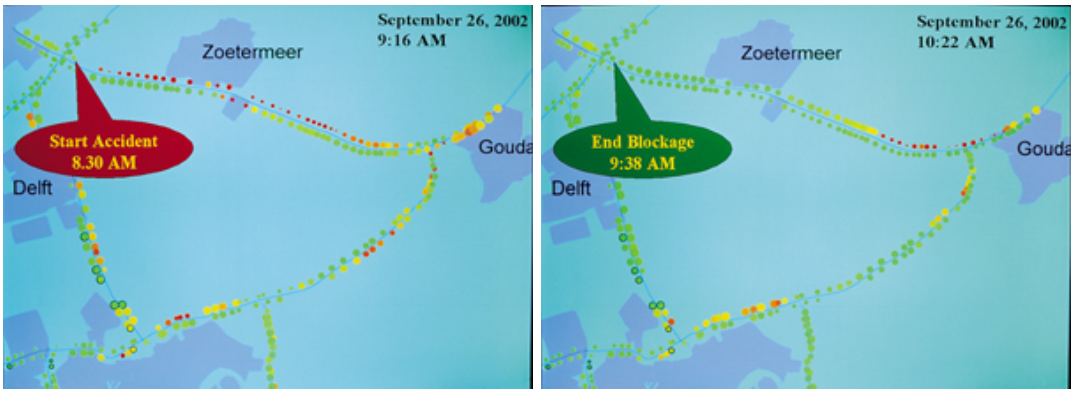
Processing of data for visualization

First, a user should select the region and the time interval for inspection. First, the files which belong to this selection are chosen. It may happen that one or more files are missing or within a file the minute data to be missing. In such case if it is possible the missing data is estimated from the surrounding data with a limit of 5 minutes.

Data visualization

After processing the data, the next step is the visualization of speed, travel time or congestion in different plots, such as: flow contour or speed contour plot (see Figure 4.3 [3]), travel time plot, travel speed plot, etc. The visualization in Figure 4.3 shows that the bottleneck is situated around the km 82-84 although a shock wave is coming from a downstream section around 17:30. Apparently it can be also noticed that between km 71.8 and 77.1 there was no data and the interpolation is not smooth. Moreover, at km 68.8 the speed is far too high and this means that recordings are wrong for that interval.

4.4 RegioLab Delft



(a) (b)

Figure 4.4: RegioLab Delft

RegioLab Delft stores and analyzes traffic information from the surrounding area (South Holland). The traffic data which they collect is primarily intended for traffic research. They have a very large database that contains minute traffic data from the ILDs and variable message signs on the highways from South Holland. The access to this data is restricted to members from the RegioLab Delft project.

Every minute of the day the number of cars passing are registered and their average speed is calculated (at the ILDs). This information with the date and the time of the measurement is stored in a database and available through, at RegioLab. The reliability of the data is also registered. This parameter usually indicates if the measurements can be trusted.

The information displayed on the website from RegioLab is only accessible for authorized users. The data that the site includes has the following parameters:

- The hexadecimal code of the measurement instrument (HC).
- The place of the instrument (PL) in meters.

- The direction (DI) (R for right and L for left).
- The number of road lanes (NL).
- The measurement date (MD) in the format *yyyymmdd*.
- The measurement time (MT) in *hh:mm*.
- Indication of the reliability of the data (IR) (where "y" stands for *yes* and "n" for *no*).
- The number of cars (NC).
- The average speed (AS) in kilometers/hour.

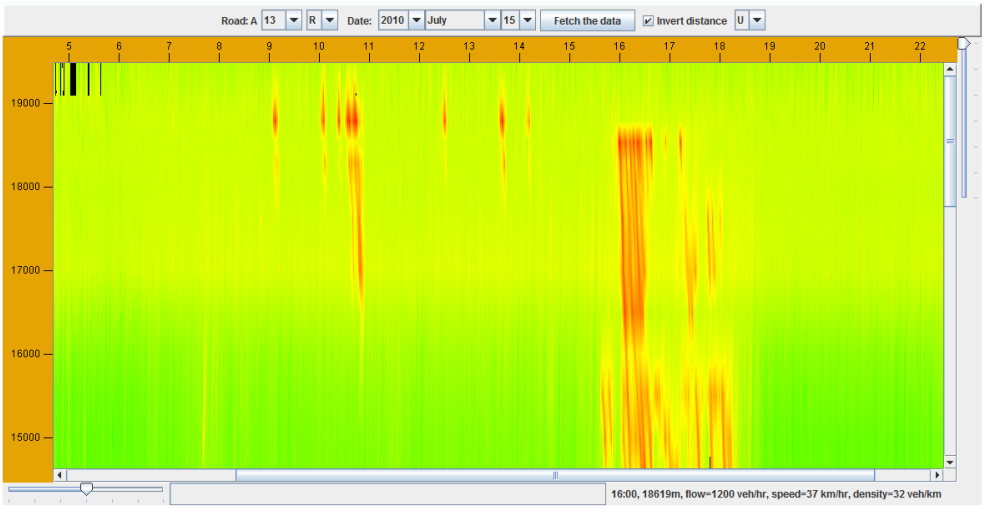


Figure 4.5: Exploring shock waves from RegioLab

One application that the RegioLab website offers the users to explore is a plot of the shock waves from the traffic in the surrounding area. It is a time, distance and speed graph. The users can select a highway in the neighborhood of Delft and a date. After fetching the data from the server (which may take up to 60 seconds) the plot appears as showed in Figure 4.5. The vertical axis in this example corresponds to distance from The Hague to Rotterdam on A13R (1 pixel is 100m). The driving direction is towards Rotterdam. The horizontal axis corresponds to the time of day, 1 pixel is 1 minute and the right of the image corresponds to 23:59. The color of the pixels indicates *observed speed* (averaged over all lanes of the roadway). Clever interpolation by using a reasonable estimation for the shock wave speed is applied for points between the detectors. Low speeds are indicated in red, high speeds in green. Shock waves are nicely visible in the morning peak and even more pronounced in the afternoon peak. The 80 km/hour zone along the last few km of this highway is nicely visible as a yellow band along the bottom of the graph. Missing data values that could not be estimated are shown in black.

The black dot marked on the graph in Figure 4.5 corresponds to an average speed of 37 km/h, at 16:00 while the vehicles flow is 1200 vehicles/hr and the density 32 veh/km.

4.5 ANWB

ANWB is one of the services which attempts to offer a *live traffic* update for the highways network in The Netherlands. The application is using data from the monitoring system. It shows real life graphical information about the bottlenecks on the highway network by giving an estimate of the

Figure 4.6: Screen shot from ANWB

current average speeds (see Figure 4.6). This traffic information is available 24 hours a day on their website and is free.

ANWB is used by people who want to estimate which would be the best road to take given a certain time of the day but it is also used with research purposes. The main output from ANWB is represented by a map which is colored depending on the traveling speed. If there is a congestion along a road this will appear to be closer to red. The roads are shown in colors ranging from from completely white to completely red. The colors indicate the congestion level on all lanes in one direction. The legend standing by the map shows an estimated current speed for different intervals of colors. Based on the legend users can judge which the estimated speed. Along with this map there also some text files provided which give for congested roads the number of congested kilometers and an estimate of the average speed.

Figure 4.6 shows an example of the ANWB traffic model at 17:15 on Wednesday for the Randstad region.

4.6 Traffic data in our model

In order to develop the prediction model of our system and eventually to test the routing algorithm, we used real traffic data. This section aims to describe the source of the data that was needed.

4.6.1 Initial historical data acquisition

The historical data that we integrated in the developed prototype was collected from the ANWB website. Why did we choose this source? The main databases collected by the Ministry of Transport in The Netherlands are not open source. It is rather complex to obtain such a database just for academic purposes such as a master thesis. In order to develop our system we also want to use just open source tools/databases and avoid the violance of copyrights. But this not the only reason, because such a database has a considerable complexity and size. Nonetheless, due to legal issues, increased complexity and time constraints we chose to use data collected in our academic environment in order to develop a proof of the concept.

The file that we used to fill in our database was built by collecting data from the ANWB website by R. Sondak [51]. The traffic data was collected each 10 minutes for a couple of weeks for a roadway network that comprised the highways and a few national roads from the country. For each road the traveling time was extracted from the text files that are offered by ANWB.

Data structure. All data was organized in an Excel file. For each road the following information was stored:

- Name of the road (such as A1, A2, etc.);
- Names of the intersections bounding the road;
- Length of the road in kilometers;
- Maximum speed allowed on the road;
- Associated traveling time computed based on the maximum speed;
- Starting from 0:00 to 23:50 for each 10 minutes interval the added traveling time in case of congestion.

The missing data in some cases was computed by interpolation from the surrounding data. After the processing and analysis of the collected data, 4 Thursdays were chosen in order to be further used. The file in Excel needed further processings in order to be integrated in our application.

4.6.2 Data structures

In order to model a dynamic traffic assignment the first data that we need is the traffic network. The fact that the average speed varies on the roads during the day determines the use of the time intervals. Time intervals are assigned to each road and are used to estimate the traveling time for a requested route. The rest of the database is concerned with the anonymous storage of cars that are routed together with the routes. This is important from various reasons such as in case of an incident when cars that were routed on the affected link should be given an alternative.

Our prototype provides also another way to fill in the database. The map can be edited by adding nodes, roads and intervals. The nodes would have a latitude, a longitude and an id. The API from the Open Street Map allowed the retrieval of latitude and longitude for each node. This can be a useful thing in case we just want to do some tests on a smaller network.

In order to test the algorithm we integrated the data which was provided together with the historical data. This was also because the historical data was collected for a certain collection of nodes and roads. These were listed in the Excel file that provided the time intervals. The data from this file was loaded in Java in the database. More details are presented at the implementation, Chapter 9.

Chapter 5

Open Source Tools

The choice of the correct tools sometimes makes the difference between a successful and a failed project. Because of this, great care must be exercised when choosing the needed tools during the development of the project. We had to make several decisions about what tools to use for different purposes: the programming language, the GIS framework, database technology, targeted platforms and development environment.

For each of these we considered various options, and we will describe in the following sections the advantages and disadvantages of each option. From this we decide which is the best solution for each topic.

Section 5.1 presents the Open Street Map infrastructure together with the reasons which determined us to choose it for the GIS framework. We also describe briefly Google Maps which represented an alternative to OSM and mention the inconveniences of using it in our prototype.

The programming language that we chose is Java and Section 5.2 describes the main aspects that we took into account when choosing the programming language. We also present the features of Java which determined us to choose it. Moving on, we present the development environment that we used, namely NetBeans, in Section 5.3. We emphasize which were the most relevant features of NetBeans that we used in our project.

The last section presents the J2ME technology that we used in order to develop the mobile client.

5.1 GIS framework - Open Street Map (OSM)

When choosing a GIS framework several aspects had to be taken into account:

- Data availability - as this is a master thesis project, complex/proprietary solutions would be inadequate as companies are really protective with their products;
- Ease of modification;
- Multiple programming language libraries for various environments.

Considering these requirements we focused on two major frameworks that allow the developers to interact with the framework.

Google Maps

The advantage is that it is very easy to integrate within a web application. This means desktop computers but also smart phones. Still, this can be considered also a disadvantage as because of the easiness, the modification possibilities are rather limited and to obtain significant improvements, hacks have to be developed around the core possibilities.

Also, the data is proprietary, so any further extension is more difficult to be make.

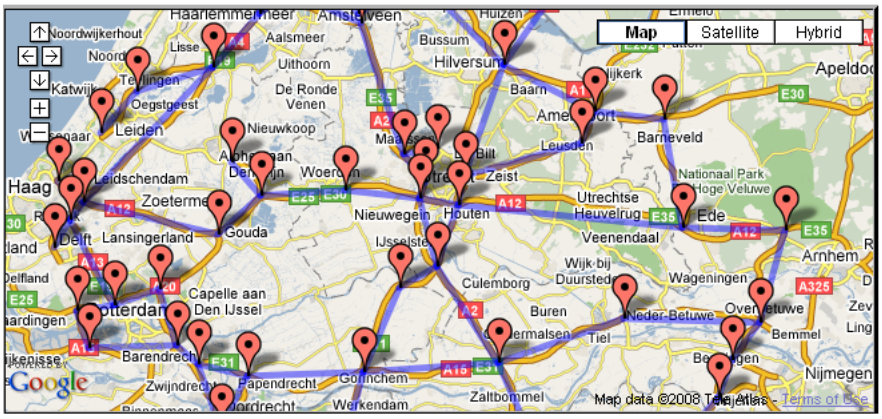
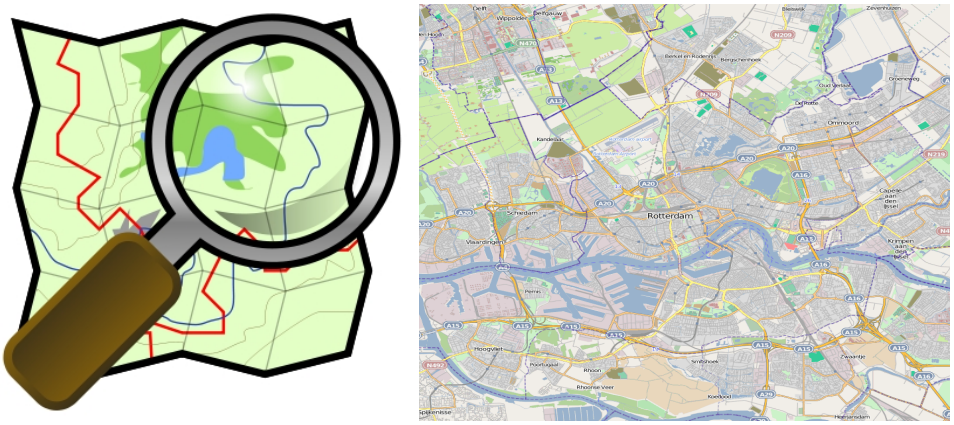


Figure 5.1: Traffic network displayed in Google Map

The main programming language used for Google Maps is Javascript, which is a bit more difficult in debugging and controlling the execution.

These ideas were gathered while working on a first prototype for this project. A picture with the interface of the prototype is presented in Figure 5.1.

Open Street Map



(a) OSM Logo (b) Map of Rotterdam

Figure 5.2: Open Street Map

OpenStreetMap (OSM) is a collaborative project to create a *free editable* map of the world. OSM follows a similar concept as Wikipedia does, but for maps and other geographical facts. An important fact is that the OSM data does not resume to streets and roads. Anybody can gather location data across the globe from a variety of sources such as recordings from GPS devices, from free satellite imagery or simply from knowing an area very well, for example because they live there. This information then gets uploaded to OSM's central database from where it can be further modified, corrected and enriched by anyone who notices missing facts or errors about the area.

OSM creates and provides *free geographic data* such as street maps to anyone who wants to use them. The OSM project was started because most maps that people think of as *free* actually

Figure 5.3: Roads and intersections displayed in OSM for the highways in The Netherlands

have legal or technical restrictions on their use, holding back people from using them in creative, productive, or unexpected ways.

Libraries to access the resources provided by the project are available for multiple languages and purposes. As an example, several rendering libraries exist (in Javascript, Python, C and Java) and also several editing clients that allow to interact with the data.

Usage of Open Street Map in our project

In order to implement the *graphical user interface* of the system and to construct an initial database out of intersections and highways from The Netherlands, we embedded an OpenStreetMap map viewer in the application. This was a Java panel which allowed several listeners and functions to be redefined. Due to the modular design of Swing component library, the integration was an easy task.

The API of the OSM viewer provides a number of utilities for manipulating maps, allowing us to construct a robust user interface for our system. We were able to:

- Create the intersections for the traffic network graph by clicking on the map.
- Create directed links between two nodes - and we assume that all the roads are straight.
- Calculate distances in kilometers between nodes by using their latitude and longitude.
- Relate geographical coordinates to plane coordinates on the map.
- Design (in different colors) paths on the map, parallel to the main roads in order to display a requested route.

An example of the OSM integrated in our application is presented in Figure 5.3. The intersections and the roads from the database are displayed on the map.

5.2 Programming language (Java)

When choosing the programming language the following aspects were considered:

- Portability of the application: one important requirement of the project is to develop an application which can be easily used on different computer architectures and operating systems and mobile phones.
- Simple, yet expressive: a relative simple object oriented programming language was desired, given the time constraints and the purpose of the project.
- Programming experience: important because of the complexity and multitude of components in the application.

Any of the main stream programming languages could have been chosen to implement the prototype. In this section we will present the features that made Java the language that was chosen. In short, we choose Java due to the fact that relies on very well known concepts like object oriented programming and we had good programming experience in Java.

Java enables developers to write software on one platform and run it on another one. "Write once, run anywhere" (WORA) is a slogan created by Sun Microsystems to illustrate the cross-platform benefits of the Java language. This means that it is platform independent, thus it has a good portability.

It is also possible to develop server side applications using exactly the same code as in a client application. Combining applications and services using the Java language in order to create highly customized applications or services is a much easier task as all components will use the same language.

Java also allows writing powerful and efficient applications for mobile phones or any other device with a digital heartbeat.

Another advantage is that Java is a quite simple object-oriented programming language. It was designed to be easy to use, debug, compile and learn in compare to other programming languages. It is a robust and reliable language as Java compilers are able to detect many problems that would first show up at runtime in other languages.

Moreover, Java is multithreaded, a necessary thing especially in visual and networking applications. In the project this was used for displaying the map tiles in both clients, desktop and mobile while running other processes.

The alternative to Java was C#, due to previous programming experience with it. The goal of C# and the .NET platform was to shorten development time by freeing the developer from worrying about several low level plumbing issues such as memory management, type safety issues, building low level libraries, etc. thus allowing developers to actually spend their time and energy working on their application and business logic instead. Still, it works on less platforms (ex: mobile platforms) than the Java language.

5.3 Development environment (NetBeans IDE)

In order to develop Java applications for desktops, we had various possibilities for an Integrated Development Environment (IDE). The main things which influenced the choice of an IDE were the following:

- Suitability with the requirements of the project. By this we refer to the GUI development. The IDE should facilitate GUI development. Also managing a database from the IDE was considered a plus.

- Experience with various IDE.

Given these aspects and the proposed architecture and design of our system we chose NetBeans as development environment. NetBeans facilitates the development of an interactive GUI, server applications, manages databases, allows building of mobile application and desktop applications in an integrated way. Most relevant features of NetBeans that we used in our project are the following:

- It has a powerful GUI builder.
- Database Integration: NetBeans provides a CRUD (create, read, update, delete) application shell and it facilitates the integration of a database in the application.
- Provides a visual design editor with end-to-end support for enterprise applications and J2ME development.

An alternative represented the use of Eclipse. But NetBeans has a GUI builder which was extremely helpful for rapidly creating the Java application. Also the plugin system of Eclipse makes more difficult to have a working environment 'out-of-the-box'.

5.4 Java 2 Micro Edition (J2ME)

In order to develop the mobile client of our prototype we considered important the following:

- Portability.
- Support in multiple devices. Given the lack of experience for programming a mobile application, a popular environment for development would assure the fact that it has a strong background and plenty of feedback.
- Basic GUI library.

Java 2 Micro Edition (J2ME) provides a robust and flexible environment for applications running on mobile or other embedded devices such as *mobile phones,* or *personal digital assistants* (PDAs). J2ME includes flexible user interfaces and built-in network protocols. An important advantage of the applications developed with J2Me is that they are portable across many devices and supported by the major phone manufacturers such as Motorola, Nokia, Samsung, Sony Ericsson, Blackberry and others.

J2ME combines a resource-constrained JVM and a set of Java APIs for developing applications for mobile devices.

We will provide some information about creating J2ME applications, also known as MIDlets. Due to the limited size of mobile devices with regard to memory and resource availability, J2ME defines a limited version of the JVM as well. Therefore, J2ME combines a resource constrained JVM and a set of Java APIs for developing applications for mobile devices.

The most popular profile and configuration that Sun provides are the Mobile Information Device Profile (MIDP) and Connected Limited Device Configuration (CLDC), respectively. As the name suggests, CLDC is for devices with limited configurations. Consequently, the JVM that it provides is very limited and supports only a small number of traditional Java classes. The MID profile complements the CLDC configuration very well because it minimizes both the memory and power required for limited devices. It provides the basic API that is used for creating application for these devices.

Smart, usability-focused design and the Java platform's built-in execution model give J2ME applications significant performance and security advantages over other similar applications.

J2ME is still the most popular mobile development platform. This is a great advantage because many applications have been done by using the J2ME developing tool and there is a strong background. Besides this, we had more reasons to choose J2ME for developing our mobile client:

- It standardizes the use of Java for small or memory constrained devices and Java is available on most mobile phones (excluding iPhone).
- Portable applications.
- J2ME has graphical user interfaces.
- J2ME has the ability to function off-line out of wireless coverage, peer-to-peer networking.
- HTTP connections facilitate the phone-to-server communications;
- There are no licensing expenses needed for the SDK, which means that anyone can create an application and market it.

Chapter 6

Conceptual Design

An important step in this thesis was to build the conceptual design of the Personal Advanced Traveler Assistant (PTA). In this chapter we aim to describe the main features of PTA together with its main components.

In Section 6.1 we begin by presenting the main characteristics of PTA. A relevant feature is that PTA would first assure a user equilibrium solution determined by the available capacity of the network. Moreover, PTA is an intelligent agent because it detects the user's profile, combines it with his schedule and suggests him the best route.

In order to develop a usable system for everybody, that has an easy and user-friendly interface we aim to have a human-centered approach in the design of PTA. The first step in this direction was to survey the preferences of potential users regarding various types of interfaces and routing preferences. Thus, Section 6.3 presents the user survey that we did in order to get more insight into users expectations from a PTA. More details about the features underlying a human-centered design of PTA are presented in Section 6.2.

In Section 6.4 we present the detailed design of the system. Due to the fact that it connects multiple servers and clients through the network, it is a distributed system. The components of the system are described along with a detailed presentation of the object-orientated model for the central server that we aim to implement, the database schema and a mock-up for the GUI of the desktop application and mobile phone clients.

6.1 Personal advanced Traveling Assistant (PTA)

PTA gives the traveler routing advices during his trip starting from departure to destination. The traveler will benefit of the best available solution according to his preferences at the departure time. If unexpected events occur, modifications during the trip.

PTA should distribute the traffic in the network so that it satisfies the preferences of the users by taking into account the availability in the network. The system will use continuously updated traffic flow information. This information would be available from the GPS-equipped mobile phones of the users. Given that the system knows which are the route requests and the routes assigned already to drivers it can give a prediction of the traveling times on the roads in the future. This can be done by training a neural network on the relation between various traffic parameters such as the traffic stream and the traveling time.

If we return to the individual routes assignment, in case of an incident/road work the system informs the traveler on the delays and best alternative solutions. The driver will be also informed on the traveling time associated to the recommended alternative, the types of roads and eventually the advantages/disadvantages.

As we already mentioned, PTA connects to the users by a hand held device. This can be a smart phone, a routing device or a PDA.

Agent with Learning Capability

PTA is addressed to a large variety of persons. People usually have different preferences regarding the route they want to travel or the types of roads etc. Some would like to travel on the fastest or cheapest roads while others might prefer to drive on shorter distances but with lower speed.

The driver's behavior is an important issue that PTA has to consider.

In time, PTA will have the *learning capability* to detect the user profile and use it accordingly. This can be achieved by using various features (such as timetable, agenda, habits, preferred route profile) and a supervised learning method which would analyze them and recognize patterns. For example, based on the user's agenda PTA can first recommend the route that fits to his schedule. Let us consider a user with a busy life who always chooses the shortest time for his trip regardless of the cost or distance he needs to travel. PTA will distinguish this pattern and assign him a specific profile. Other people may enjoy a relaxed trip, especially retired persons who do not have to hurry and would rather avoid the turbulence of the highways. Or students who might prefer a low budget trip (avoid toll charges).

However, an important thing is that it connects to the agenda of the user so that it knows his schedule. For example, it learns the route home every day after work. When switched on at the appropriate hour, it would firstly recommend the best route for going home. In this way PTA prevents people from wasting unnecessary time. If the roads are highly congested it can include restaurants suggestions, shops or even propose an attraction park (knowing that there is nothing important in his agenda). If the user has a meeting PTA shows him at which hour he should leave his current location in order to arrive in time, depending on the congestions in the network. It would also check for a parking place and if applicable, mention how much it costs.

Intelligent Traveling Assistant

PTA is an intelligent traveling assistant because it continuously advice the traveler about the best route to follow until destination. Depending on the delays that might occur on the route it computes alternatives and suggests best solution in accordance with the user's profile and his schedule. Alternative routes can be also generated when requested by the users.

We name PTA an intelligent assistant also because it collects and computes a future estimation of the traveling time.

The system would interact with the user in a multi modal modality depending on the place and situation. For example, in a noisy environment it would preferably use the graphical interface for data input whereas in a different setting it would use the speech recognition. So, PTA has a smart interface integrated in a human-centered design. Mode details about the multi-modal interaction and the smart interface are presented in Section 6.1, along with users preferences detected in a user survey primarily related to the UI.

PTA Environment

PTA is a distributed system that links the users to the central server. All components are connected through Internet. Users are connected to the server but they can also communicate among themselves by using ad hoc wireless networks. A possibility to do this is by using the wireless network between light poles on the highways. In Table 6.1 we list the advantages and disadvantages of a distributes system. Based on this and on the fact that we wanted to have the data processings separated from the interface to the users we chose PTA to be a distributes system.

The advantages of PTA is that if the system becomes a centralized one (because of network problems) it still manages by connecting to the other cars in order to get the information it needs.

Figure 6.1 depicts the environment of PTA. The main server has to be connected to the roads and nodes database, to the historical database and to an incidents database. As we already mentioned, it is important for PTA to benefit of live traffic information. The system is designed in such a way that it uses the information from the vehicles that already exist in the network. The GPS-equipped mobile phones report their positions at fixed time intervals. Moreover, the routes of the vehicles are supervised, meaning that the system knows the origin, the destination and the

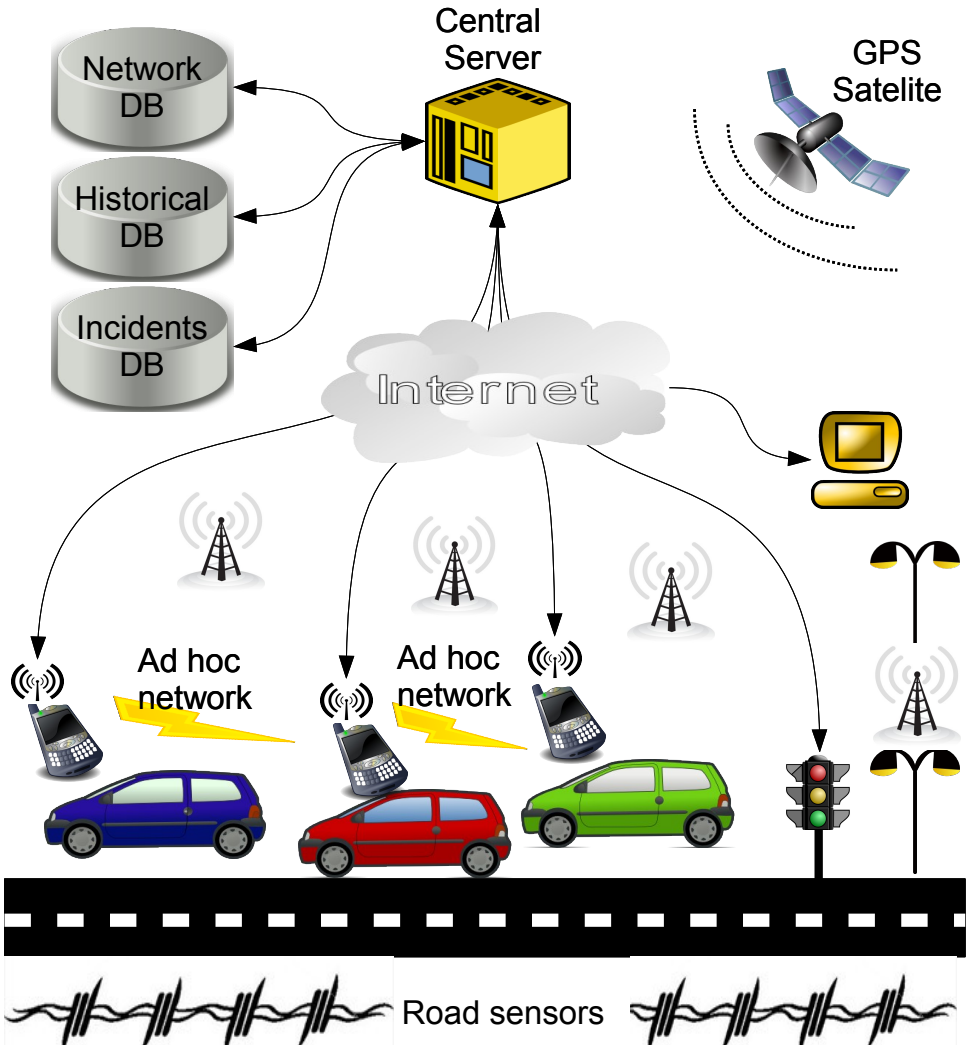


Figure 6.1: Environment of PTA

Table 6.1: Advantages and disadvantages of a distributed system

Advantages	*Disadvantages*
Scalability	Difficult to keep all system components updated
Inherent distribution	Network saturation, loss of transmissions
Reliability: If one machine crashes, the system as a whole can still survive. Higher availability and improved reliability.	Security: easy access also applies to secret data
Incremental growth: Computing power can be added in small increments. Modular expandability	

departure time of each route. In this way the traveling time of the traffic flow in the future can be estimated.

6.2 Human-Centered Design

We aim to develop a system that is a usable system, by everybody, as the potential users of PTA would be really diverse. Why is this feature that important? Usability is now widely recognized as critical to the success of an interactive system or product [52]. There are many poorly designed and unusable systems on the market that people either find difficult to use or use them wrong. They can even get frustrated and stop using them. The outcome is harmful to the reputation of the company who deployed the system. Even more, the system would not fulfill his purpose, namely to improve peoples lives.

We need a human-centered approach as we attempt to have a system that would be used *easily* and *with confidence* by most of the people. Key principles of the human-centered design (HCD) that we integrate in our design are the following:

- *An active involvement of the users.* For our system we collected suggestions from potential users in order to develop the design of PTA and especially the interface. With this scope we did a user survey that is presented in Section 6.3. We also informed on state of the art features of such applications with respect to features, guidelines, design and user tests. There was as well a demo session of our prototype which allowed us to receive feedback about the system from a wide diversity of persons.

- *Allow both the user and the system to take decisions without giving any one of these entities the ultimate control over the result.* This is a sensitive issue because on the one hand, most users wish to have a PTA which does everything by himself, offers the best solution, that is very simple to use and never crashes. On the other hand, it is undesirable that PTA gives a vary bad solution. This can happen because of various issues, like a malfunctioning sensor or a bad weather forecast. That is why it is desired to let both the user and PTA take decisions without leaving any of these two entities the ultimate control.

- *Clearly state the requirements of the system and evaluate the system.* Most important requirements of the system we plan to develop, the design we modeled and the prototype we implemented were presented at Section 1.6. In the end, we aim to evaluate the system based on these requirements (Figure 6.2).

Following Figure 6.2 we see that a crucial aspect is to understand the *context of use*. This was our first step towards designing a smart interface. Main questions that raised are the following: why do we develop the system, which are the objectives, who are the users and what they need from the system, which are the technical constraints, what type of device will be used and what GUI (graphical user interface) should be designed.

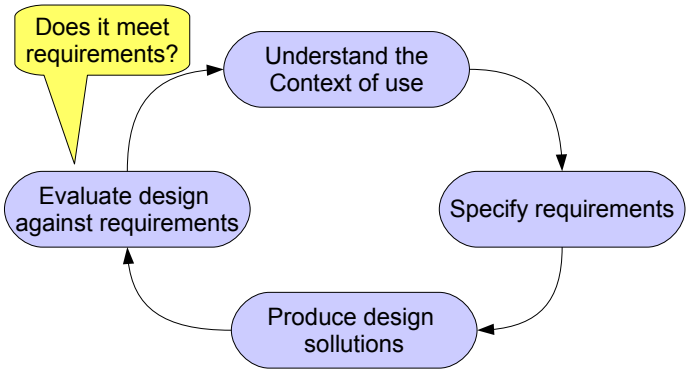


Figure 6.2: Human-centered design circle

An important part of our system with regard to the human-centered approach is the interface to the users. We aim to design a smart interface that allows the user to interact in different manners with the system, depending on the case. More specifically, we refer to various possible input and output modalities (presented next) of the system.

Input and output modalities for the interface

The system interacts with the user depending on his profile or the situation in which it is used. The user has to be able to choose among several modalities the one that fits him best. He should also be able to switch them. In the following list we present some input-output communication modalities between the user and the PTA along with their description and advantages or disadvantages:

- *Speech recognition.* This modality assumes that the PTA has a speech recognition module integrated. On highly dense roads the PTA should interact with the user through *speech*. In this type of situations or in the cities the PTA has to be as concise as possible in order to not disturb the driver too much. The user should be able to communicate the PTA by saying the destination address. The system would use a solution that fits user's preferences.
- *Graphical Interface.* PTA offers a graphical interface for both desktop and mobile clients. We assume that the user is most of the time connected to the system through his mobile phone but he might use the system also at home on his desktop. The GUI of the mobile phone should allow the user to *point on the map* for showing his destination. He may also *insert the address* or postal code of his destination. The map would allow him to check his route, to zoom in and out or to move it. With regard to the *map orientation* we aim to let the driver choose the one which suits him best. Some GPS devices present the map having the north in front. This may confuse some drivers.
- *Use of the driver profile.* The PTA may use the driver's profile as input modality by an association of his trips to his agenda.

6.3 User Survey

The user survey is an approach to better understand which are the expectations of people from a personal traveling assistant. For this project it represents an important step towards a human-centered design. The usability of the system depends on how easy is for people to interact with the system. This has to be done in an intuitive way, so that users like the interaction and find it natural. We informed ourselves on state of the art applications, systems and demands so that we

figured out which could be the important features for a PTA. Therefore, the survey shouldn't be only a way to discover new ideas but also a way to confirm or not our expectations.

In this section we aim to present the dimensions of the survey (what we surveyed), the features of the users who participated at the survey, an assessment of the results and the extent to which answers met or not our expectations.

The concept of PTA was introduced at the beginning of the survey through a short summary so that people understand which are its features and capabilities.

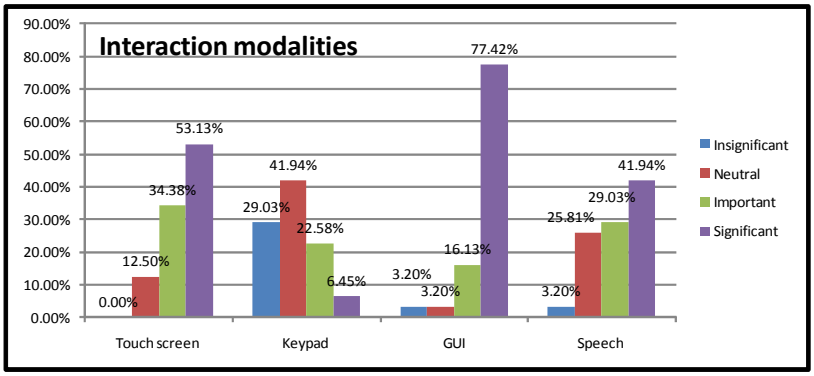


Figure 6.3: Importance of interaction modalities with PTA for the respondents

The main things that were surveyed are the interaction modalities with PTA but also the routing possibilities in case of unexpected events. The interaction modalities refer to the input, output interface but also to the ways in which PTA would alert the user and change the route in case of incidents. Besides this we also ask the users which is their opinion on certain features of the GUI of PTA.

We used a sample of 32 potential users of PTA with an average age of 25 years, starting from 19 up to 35. All of them are regular drivers. Within the group there was a female percentage of 31.3% and a male percentage of 68.8%.

6.3.1 Results

Interaction modalities

With regard to the interaction modalities the survey checked which modalities would be of greater importance for the users. Moreover, there were also questions on which modality they prefer to use when entering their destination. Other questions related to the modality in which PTA can announce a route change or transmit an information.

The first question referred to the interaction modalities that we consider important for the interface of PTA. The respondents had to mention on a scale varying from 1 to 4 (Insignificant, Neutral, Important, Really significant) how significant they find each of the following: a touch screen, a keypad, a GUI and the speech recognition. Figure 6.3 shows a graphical representation of the users preferences towards the 4 modalities aforementioned. As a conclusion, a GUI was found really significant by 76.67% of the subjects, a touch screen was also very important for 51.61% whereas speech recognition was significant for 40% of the users and a keypad was seen neutral by 43.3% but important for 22.5%.

In case an accident happens on the route, PTA computes the alternatives and changes the route. With regard to this situation, subjects had to answer whether they would like PTA to write

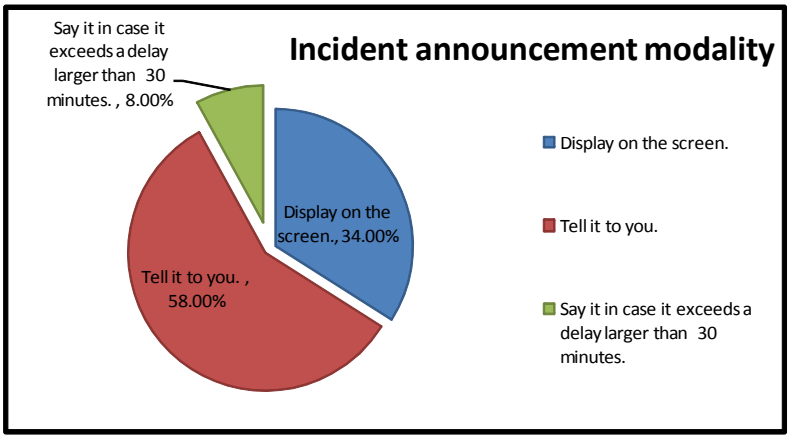


Figure 6.4: Modality to announce an accident on the route

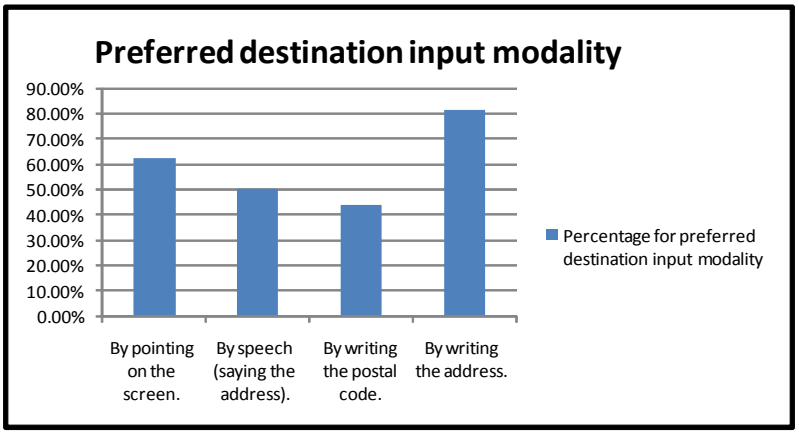


Figure 6.5: Preferred modality to communicate the destination to PTA

it on the screen, say it or say it only if the delay exceeds 30 minutes. It may be annoying to have your traveling assistant always announcing which is the situation. Figure 6.4 shows a pie-chart representing users options. We notice that most of the users, that is 58% would like to hear from their PTA if anything changes. Nevertheless, 34% would like just to see it displayed in the screen.

When asked how many times PTA should repeat the route change, 60% of the subject would prefer to hear it twice while 31.25% three times. This feature should depend on the noise level as only sometimes it might be necessary to hear the same information more times. However, in the same survey people complained about their current GPS device repeating too many times the direction changes.

When PTA has to transmit information 60% of the subjects prefer the speech modality instead of the graphical interface (displaying on the map, showing on the screen etc.).

In order to enter the destination location for PTA we proposed several modalities such as: pointing on the screen, by using speech (saying the address), by writing the postal code or by

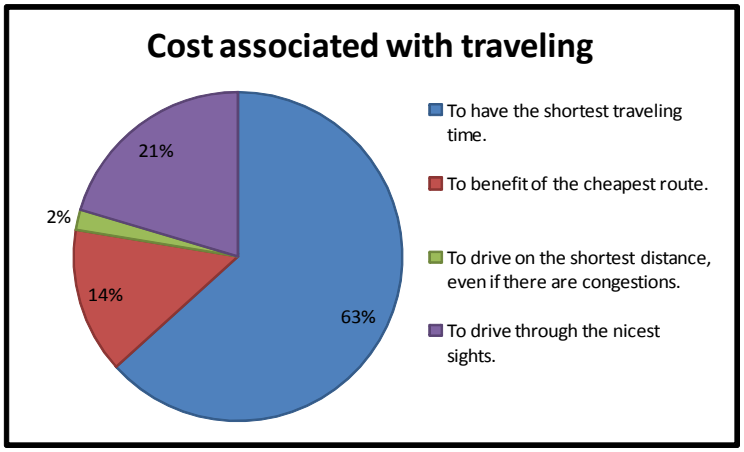


Figure 6.6: Most essential for a route request

writing the address. For this question the subjects could choose more than one modality. As we can see in Figure 6.5 most users are in favor of writing the address, almost 80% while 61% would like to point on the screen. There were less persons choosing the speech recognition modality. A reason for this could be that people do not trust a speech recognition system because if there is any noise around it gets more difficult to transmit the right information.

Moreover, in order to ease the search of a destination, in case the user does not know the complete address, an important thing would be a generalized search button. In this way people would easily find places like a specific supermarket, cinema, restaurant, etc. without having to follow a special menu. This is not the case of current GPS devices and it is another drawback.

Routing preferences

With respect to the users routing preferences we were mostly interested in how the users would like PTA to announce a route change and what is the cost that they usually associate with their trips. We also wanted to find out whether people would like to have a profile done by the PTA and that their route request is assigned based on their profile and agenda. With respect to the latter issue 87.5% of the surveyed persons had a positive answer.

The cost people usually associate with their trips refers to what is most essential for them when requesting a route. With respect to this issue the subjects could choose between the following: the shortest traveling time, the cheapest route, the shortest distance (even if there are congestions) or to drive through the nicest sights. As expected, most people prefer the shortest time (63%) whereas just 2% prefer the shortest distance. A pie-chart of the percentage of people who rated one of the 4 associated costs is depicted in Figure 6.6.

In case of accidents and long delays most of the subjects would like to be guided on an alternative route (87.5%) instead of another activity recommended. Nevertheless, in case of delays that exceed one hour, 57% of the subjects would like to have a different activity suggested.

GUI

With regard to the GUI of PTA some of the improvements that we suggested are the following:

- The possibility to show the level of congestion on the roads by different colors;

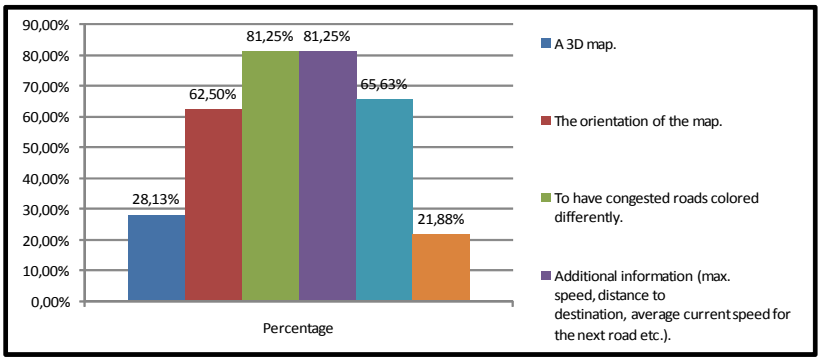


Figure 6.7: Assessment of important features for the GUI of PTA

- Show the actual average traveling speed for the next roads on the route;
- Display a 3D map (this would help people recognize the place where they are easier);
- The possibility to select the map orientation (some current GPS devices always show the north in front and this might confuse the users);
- The possibility to visualize additional information such as the remaining time or distance, speed cameras etc.

When asked whether PTA should mark the accidents and the congestions on the map, 90.6% of the subjects had a positive answer, as we already expected.

Further on, an assessment of some features that might be comprised in the GUI of PTA was done. Figure 6.7 shows which are the options that the subject could choose. Multiple answers were possible. We notice that people would like to have the congested roads colored differently (81.25%) but showing the speed cameras would be also important for 61.6% of the interviewed persons.

Users recommendations for PTA

Besides the characteristics that PTA already has we asked the subjects to give a suggestion that they have for a personal traveling assistant. Some of the most interesting suggestions are the following:

- Simplicity: people would like PTA to be as simple as possible and route them from point A to B with the shortest traveling time. Still, they would like to be aware of where they are and if anything changes.
- Correlation with their agenda.
- Automatic maps and software updates.
- Possibility to call the ambulance or the police.
- Compute the complete cost of the journey including fuel, toll charges, etc.
- Correlate with the radio so that the sounds does not overlap.
- Possibility to mark things like speed cameras. If there are more users reporting one at the same location then it should be displayed for everybody.

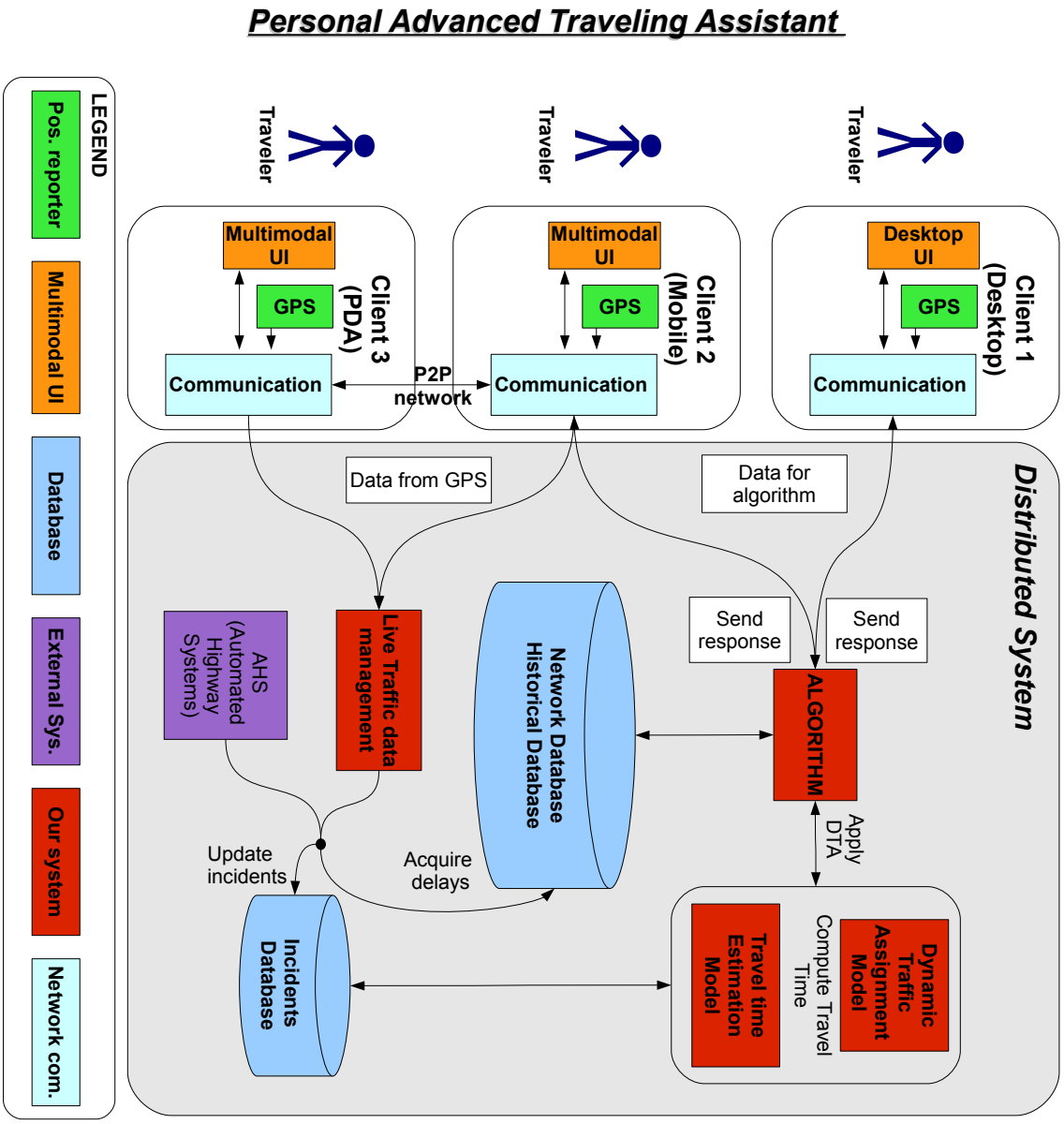


Figure 6.8: Detailed design of PTA

6.4 Detailed Design

The application that we designed represents a distributed system that connects multiple clients and servers through the network. We assume the existence of multiple clients that interact through a wireless network. All interact with each other in order to achieve the same goal. The main components of the system, as we can see in Figure 6.8 are the following:

- Data Providers. The data providers represent the traffic network database (roads and intersections along with all the information), the historical and current traffic information along

Table 6.2: Advantages and Disadvantages of a client-server communication

Advantages	**Theoretical disadvantages**
Centralization - access, resources and data security are controlled through the server	*Dependability* - when the main server goes down, operations cease
Scalability - any element can be upgraded if needed	Need to implement a communication between client and server
Flexibility - new technology can be easily integrated in the system if needed	
Inter-operability - all components (clients, network, server) work together	
Accessibility - server can be accessed remotely and across multiple platforms	
Ease of application development	

with the incidents database.

- Central Server. The central server is connected to the data providers and applies the dynamic traffic assignment model. It is also connected to the AHS (Automated Highway Systems) in order to collect traffic data. But live traffic data is gathered also from the GPS equipped mobile phones that are participating at the traffic.
- Mobile Clients. These are the mobile phones of the users. Routing requests from the mobile phones are processed on the server and the routes are sent back. Mobile phones communicate also between each other in order to have traffic updates transmitted. The mobile phones report their position at fixed time intervals so that the traffic flow can be detected.
- Desktop Clients. The desktop applications can be used by offline travelers who just inform on the traffic state.
- Connection to AHS. These are represented by the sensors in the roads which compute locally the flow and send information like average speed and number of vehicles.
- Traffic data management. The GPS equipped mobile phones send their position at fixed intervals. The traffic data management combines this current data with historical data. The travel time prediction model uses this data in order to estimate the traveling time that is sent to the algorithm.
- Smart UI. It links the user to the mobile and desktop clients.

The architecture of PTA was designed to satisfy our requirements and fulfill our goals. This design allows us to have the data providers separated from the traffic assignment and travel time estimation models from the processings of the algorithm and the UI.

This design has the advantage of flexibility because each side can be implemented using different technologies. We are also able to extend any of these components later on independently.

6.4.1 Client-Server Communication

The communication between clients and the central server is done through a client-server protocol. A server side implementation of the algorithms was chosen instead of a client side implementation due to the fact that is allows parallel requests and it is much faster.

Table 6.2 describes the advantages and the theoretical disadvantages of a client-server communication. The disadvantages listed here are mostly theoretical because from the point of view

of dependability, the application would depend anyhow on communication with peers, and any medium could be seen as unreliable. In case of a centralized server, multiple solutions exist to increase the reliability, such as:

- Multiple backup servers that can take the load if the main server goes down;
- Clients can allow the server to be unreachable for a period of time, and cache locally results.

6.4.2 Central Server (OO Model)

As the technology chosen for the server is Java we will use an UML class diagram to describe our object system. We find this an appropriate way because the class diagram is generally used to define a more detailed design of a system. The relationship or the association between the classes can be represented. Each class in the class diagram provides certain functionalities (methods). A class diagram can also be used in order to analyze the requirements in the form of a conceptual model. But at this stage we mostly use it to depict the detailed design of the OO server application of the system.

Some of the advantages of using an UML class diagram are the following:

- UML is a language used to model systems and make them readable;
- UML is a standardized language for representing the design of a system which can be easily applied across the board for all object-oriented system;
- The class diagram can be used to illustrate the classes of the system, the relationship between the objects and the attributes and operations of the classes.

Figure 6.9 represents the class diagram of the main server application. The diagram aims to represent the most relevant objects for the server that we will implement in Java. One purpose of the diagram is to show the procedures that stand between a route request from the user and the result, which is a route response. Moreover, the relationship between the data structures, the algorithm and the traffic model are presented. Because we want to represent a readable diagram it does not present all methods that would be eventually necessary.

The data structures which we use are nodes, roads, time intervals, cars and routes. A node represents an intersection and it has the following private attributes: the geographical coordinates like latitude and longitude and a name. An important method in this class is the retrieval of a node's neighbors which is needed in the algorithm. Other methods could be the retrieval of roads that come or leave from a node etc.

Roads have as private members the edges (a *from* and *to* node), a name, the length and a speed limit. The length of the road can be computed by using the coordinates of the edges. The speed measurement unit is in km/h. An important method of this object is the retrieval of time intervals for a certain time range. This is necessary for the estimation of the traveling time. Between roads and time intervals we have an aggregation relationship. Each road has time intervals assigned which represent the variation of speed during the day. An interval is defined by a time range (start time and end time) for which an average speed is assigned. This assembly represents the integration of historical data in the system. Because of the fact that we would like to extend the model and use historical data for different types of days the intervals are also defined by the interval type.

Routes have a car, an origin and destination, a departure and an arrival time associated. Methods that can be assigned to this object are the following: the retrieval of routes for a car or for a specific time interval. These can be used in order to detect the number of cars which are in the network or to re-route cars in case of incidents. The result of the routing algorithm is a route. That is why another important method is the computation of the route length and the fuel consumption. The route will be transmitted through the network to the user by using a vector of nodes, the traveling time and the distance along with other details.

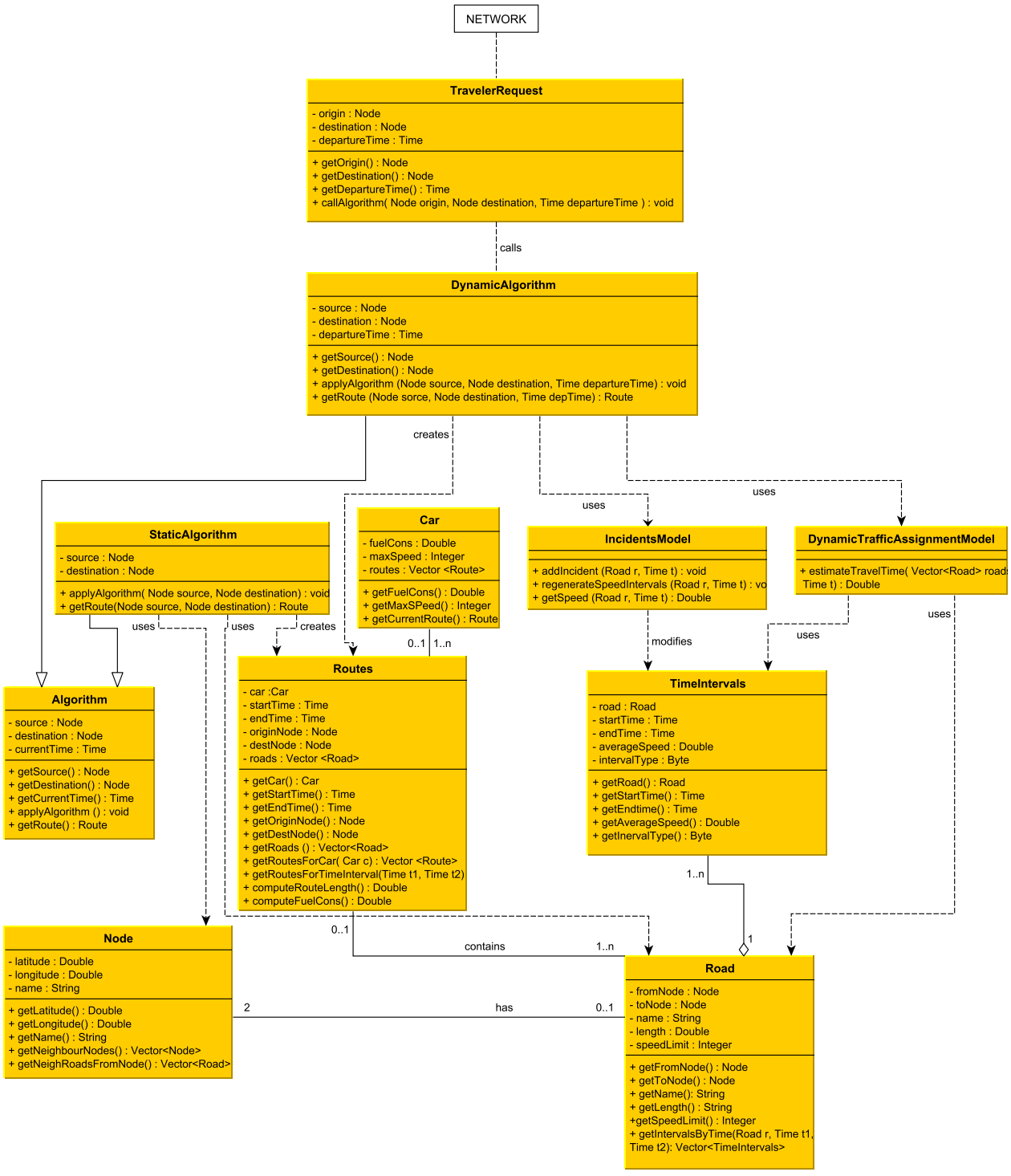


Figure 6.9: UML class diagram for the OO server application

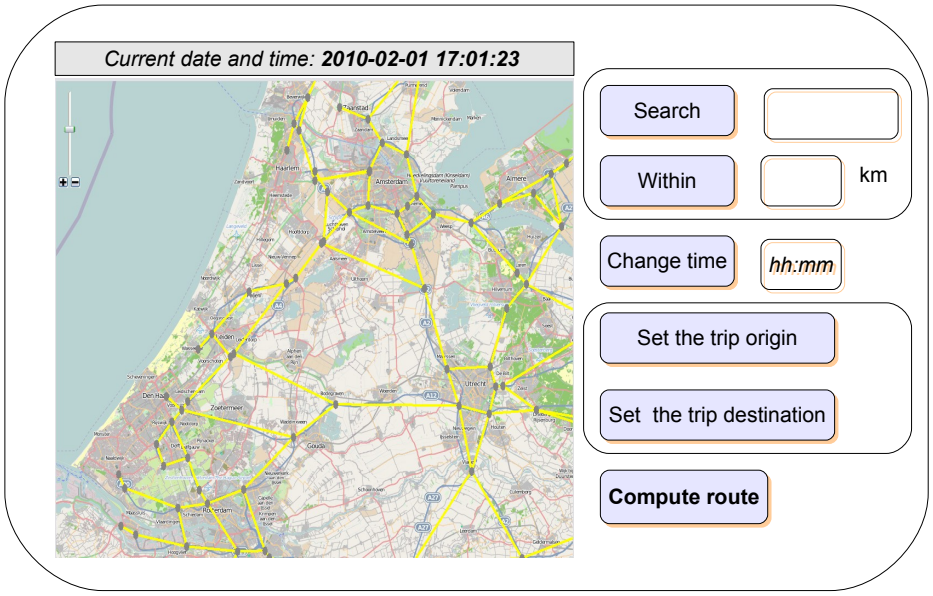


Figure 6.10: Desktop Client GUI Mock-up

A route request consists of source, destination and departure time. The input from the user may be in the form of postal code or point on the map. This is sent to the server through the network. The server will identify the closest nodes and assign the source and destination a node in the roadways network. The departure time is entered in the form *hh:mm* and sent to the network where this String is parsed. Although the type associated for it in the diagram is Time an Integer value will be used as we parse the String and compute the number of minutes.

The route request calls the algorithm. Because we aim to compare more algorithms and compute more solutions we create an interface for the algorithm. This interface will be implemented by any class that provides a routing service. The private members of this class are the source and destination nodes along with the time. The static routing algorithm uses just the origin and destination and computes the shortest route. The dynamic algorithm applies the dynamic traffic assignment model which estimates the traveling time. This is done by using the time intervals associated to roads. The travel time is dynamically computed by taking into account the time at which the car would arrive in each intersection.

6.4.3 Clients interface

In this section we aim to describe the design of the GUI that we aim to implement for the desktop and mobile client. For each client we will develop a mock-up interface based on the requirements, user survey (Section 6.3) and the user input.

Desktop Client

Given the requirements and the user survey results the interface of the desktop has to be an interactive interface that provides the user the possibility to make a route request and receive an answer based on his profile. In order to show a possibility to use the profiling feature we will use a set of profiles such as: the shortest route, the shortest traveling time, no highways, avoidance of toll charges etc. These categories should be anyway related to the user's agenda, so that the first assigned route is the one which suits best the situation. Nevertheless, in this prototype we use a profile option that stands for this.

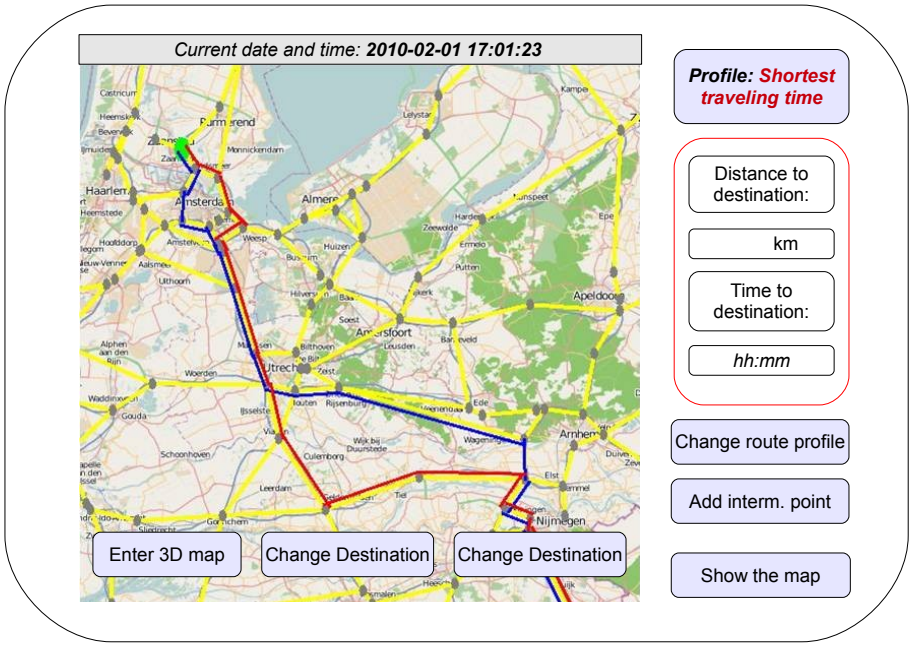


Figure 6.11: Desktop client routing GUI mock-up

In Figure 6.10 we present a mock-up of the graphical user interface of the desktop client. We aim to create an interface that is easy to use and transmits the essential information.

An indispensable component of the interface is the map. The user has be able to easily zoom in and out or move the map. In the top right corner we have a general "Search" button and a field to fill in within how many kilometers the search should extend. This should allow users to search a supermarket, or a specific parking place etc. Results might appear as dots on the map so that the user can click and set one as destination if he'd like to. In order to insert the destination the user should click on the map and press "Set Destination". The same procedure is valid for the origin of the trip, in case it is different than the current location. Next, the user has to press "Compute Route" in order to send his request. However, we assume the desktop application is used by people who are not already participating at the traffic but who want to check the traffic states at their departure. That is why it is possible to enter also a different time and origin than the current ones.

The server sends the solution which suits most to the user profile but he/she is also able to change the trip profile by using the "Change route profile" button. Figure 6.11 depicts the most important components of the routing GUI: we have the map with the route and the possibilities to change the map orientation, or to enter a 3D map. Most important information about the trip such as remaining time and distance to destination are also shown. The user is also able to insert an intermediate point.

Mobile Client

In order to describe the mobile client we present a mock-up of the graphical interface that we aim to implement in order to access the application from a mobile phone.

The first requirement of the GUI from the mobile phone is to display an interactive map which means that it should give the possibility to zoom in/out or to move it. Following this, it should be possible to enter a route request by pointing on the map for a destination location. This can be done with a touch screen or by using the keypad (entering the address or postal code). The

Figure 6.12: Mobile phone - GUI mock-up

request is sent to the server through the network where the traffic assignment model is applied. The answer from the server consists out of a route solution that can be displayed in a graphical and in a text mode. Figure 6.12 shows a possibility to display the route received from the server on the map. Next, the user should be able to access a routing menu that would allow him to add an intermediate point, search for a new place or change the route profile.

With regard to the graphical interface while driving we aim to use the natural orientation of the map, zoom in for each route and mention each direction change. Moreover, if an incident occurs on the route the device should announce the driver and re-route him.

6.4.4 Database Schema

In order to manage the data structures in our application we choose to store the data in a normalized relational database system. Figure 6.13 shows the database schema that is employed by our system. The relational database schema defines the tables, the fields in each table, and the relationships between fields and tables.

The nodes table corresponds to the intersections in the network. It has a primary key which identifies each node, of type Integer and the latitude and longitude which are Float and for which up to 4 decimals are considered. A node has also a name that is a Char of maximum 30 characters (such as "KnooppuntDiemen").

The roads table have a primary key which identifies each road and two foreign keys that represent the edges of a road. Other attributes are the name which is a Char of maximum 20 characters (such "A16"), a speed limit which is an Integer and is given in kilometers per hour (km/h) and the length, which is Float and used in kilometers.

Each road has a list of intervals and this determines that the intervals table has a foreign key pointing to the corresponding road. Other attributes of an interval are the starting and the ending time which will be considered Integer values, counting for the minutes. Another field is the average speed of the flow, a Float that represents km/h.

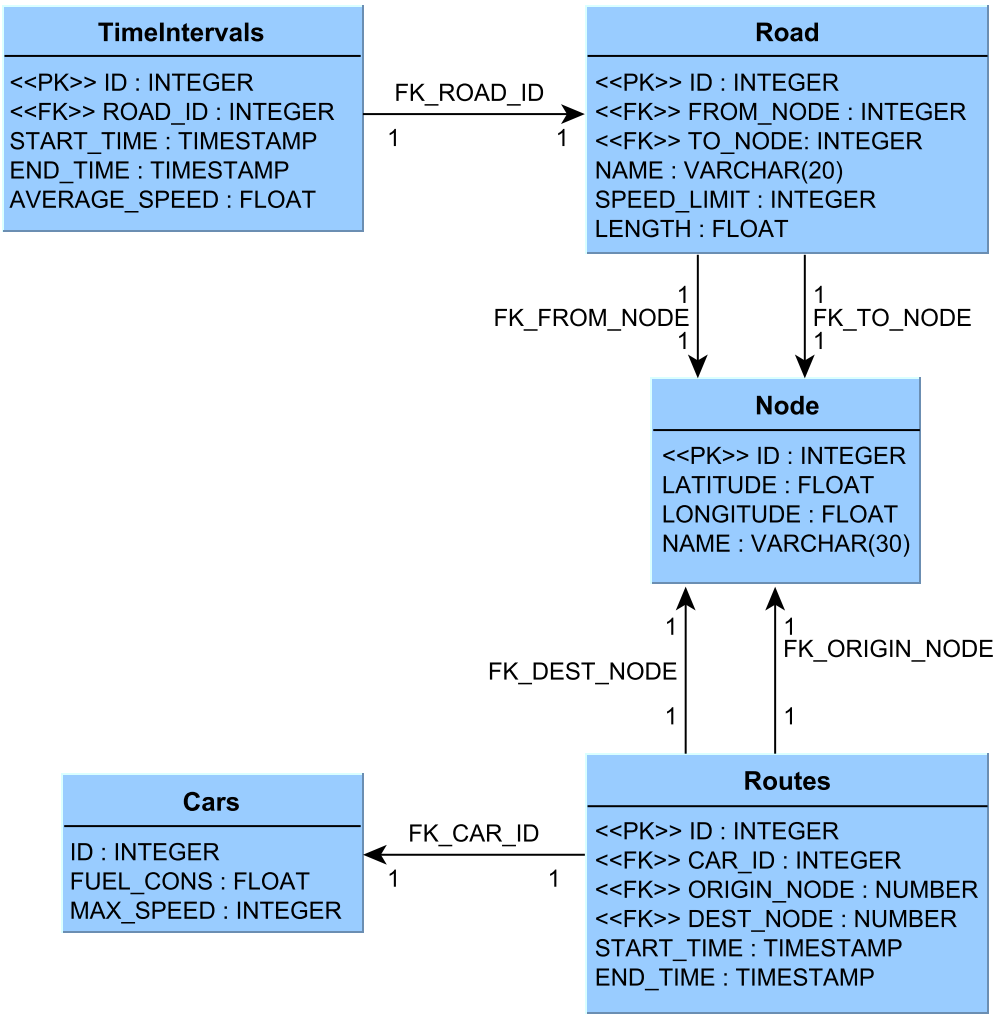


Figure 6.13: SQL database schema

The routes table has a primary key and three foreign keys: two that represent the origin and destination node and one for the car. The other two attributes correspond to departure and arrival time.

The cars table has a primary key, a fuel consumption that is given in liters for 100 kilometers and a maximum speed given in km/h.

Chapter 7

Dynamic Traffic Model Development

In this chapter we will describe our Dynamic Traffic Assignment (DTA) model and the steps that we followed to develop it. The development of a DTA model represented a basis for this thesis. The DTA model is employed by the algorithm in order to compute the routes.

We developed various models, by augmenting each model with more and more advanced processings. The last model is the most advanced but we gained valuable insight during the development process, so we will present in this chapter all models.

We discuss the things that we found most relevant for our DTA model in Section 7.1. Following this, we present the steps which we followed in order to develop a dynamic traffic assignment model. First we considered a simple model that applied a static algorithm in order to find the solution for a route request. This is presented in Section 7.2. Following that, in Section 7.3, we develop a model that uses the variation of the speed associated to each road. At this stage we simulate the dynamics of traffic by using traffic data based on our empirical experience. Moving on, Section 7.4 explains how we integrated real historical data in the model in order to have a better approximation of real life situations. Finally, in Section 7.5.2 we introduce a simple model for dealing with incidents in the network. Section 7.5.1 presents the method which we propose for creating historical traffic models for different days, forecasts etc.

7.1 Dynamic Travel Time Prediction model

The traffic prediction has to rely on a model of the reality. As we cannot model all the details due to the extensive complexity of all factors which influence traffic assignment, we will build an abstract model. This model will omit certain factors that we consider not to affect the final outcome. Where appropriate we will give arguments why those factors are omitted. Primarily, we focus on the prediction of the shortest traveling time. There can be also other metrics to evaluate traffic assignment, depending on the user's preferences. Users can be interested for example in the cheapest route but not the shortest in time. However, in our model we focus to obtain the shortest route in time based on the historical data and eventually current traffic stream updates.

We will deal with the traffic flow and traveling time from the network but we will not take into account the actual capacity of the roads or the number of lanes. With regard to incident models for instance we currently consider just a simple model. We estimate the queue that forms along and compute alternative routes.

We will also not consider for the moment the drivers behavior in our model. The drivers behavior can influence a lot the realization of traffic. There might be drivers who do not like driving with the highest speed allowed. Some might also avoid the fastest lane. However, in our model we assume that all vehicles travel with the average speed.

Dynamic models differ from their static counterparts because they represent the traffic variations over time. That is why including the temporal dimension in our traffic prediction model is necessary to represent a realistic behavior of the traffic flow. In order to develop a *dynamic travel time prediction model* for our system we established and followed the following steps:

- **Step 1**. Given the network graph and a route request we provide the shortest path from origin to destination. At this step we aim to develop a model which computes the traveling time by using the maximum speed allowed on the roads. This is also associated with the notion of a static traffic assignment model. This model is described at Section 7.2.
- **Step 2**. In real life traffic has a dynamic behavior. The speed on a road is usually associated with a variable graph. At this phase the variance of speed in time on the roads is integrated in the model. We develop a dynamic travel time prediction model that uses empirical databased on our experience. This model is described at Section 7.3.
- **Step 3.** Historical data is integrated in the model for a more effective approximation of real life situations. At this step a proof of the dynamic assignment concept is conceived. This is presented in Section 7.4 along with the modality in which the algorithm computes the traveling time.
- **Step 4**. At this stage we aim to develop an incidents model prototype. Due to time constraints and the complexity implied by the development of such a model we stick to a simple model, that can be later on improved, in order to show that it could be easily included in the system. More details are described at Section 7.5.2.
- **Step 5**. At this stage we consider to extend the model by including various historical traffic models for certain types of days. In order to do this we aim to use real traffic data in order to update the historical data. In this way the estimation of the traveling time depending on specific days, becomes more accurate. This is presented at Section 7.5.1.

7.2 Static model

Given the weighted graph $G = \{E, V\}$ that represents the traffic network we have E links, which are the roads in real life and V which stands for the set of nodes in the graph, which represent the intersections in real life. For each road in the graph we have associated

- $d_{i,j}$- the distance between node i and j.
- $S_{i,j}$- the maximum speed.

We assume that we have a route request from node *O (origin)* to node *D (destination)* and the minimum O-D path has to be assigned. We define the subgrapf $G' = \{E', V'\}$ which represents the set of roads and nodes forming the OD trip path. For each edge of G' we have the weights w_i that represent the *shortest distance*.

Given the graph G', we have A the adjacency matrix which is an $n \times n$ matrix where n is the number of nodes. If there is an edge from some node x to a node y, then the element $a_{x,y}$ is 1 (or in general the number of xy edges), otherwise it is 0.

We apply the Dijkstra shortest path algorithm and we obtain a route p - a sequence of nodes such that

$$p = \{n_1, ...n_m\}, \tag{7.1}$$

where $n_1 = v_O$, $n_m = v_D$ and $\nexists p^{'}$ such that

$$\sum_{i=1}^{n} d_{P'_i, P'_{i+1}} < \sum_{i=1}^{n} d_{P_i, P_{i+1}}. \tag{7.2}$$

The traveling time is computed based on the maximum speed allowed (Equation 7.3). This approach is also known as static routing. Most GPS devices used nowadays offer this solution for a route request.

$$T = \sum_{i=1}^{n} \frac{d_{P_i, P_{i+1}}}{S_{P_i, P_{i+1}}}. \tag{7.3}$$

7.3 Speed variance model based on empirical experience

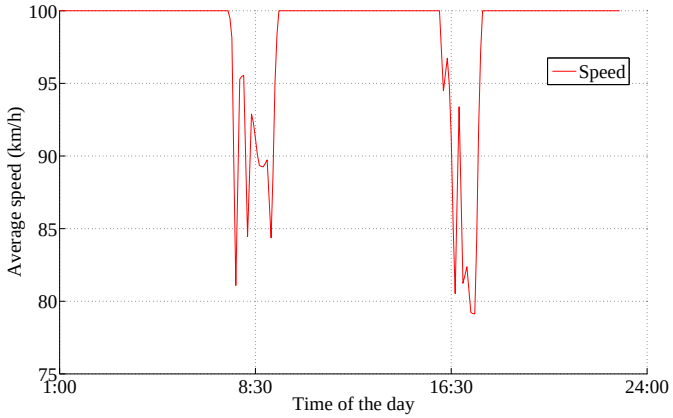


Figure 7.1: Example of a speed graph

At this stage we consider that one requirement of the dynamic assignment is the estimation of the traveling time with a low prediction error. The estimation of the traveling time in real life cannot be done based on the maximum speed because of congestions. Thus, our next step was to consider that the traveling time during a day in real life changes.

For this model the weights of the graph w_i for each road change during a day. Due to the dynamic behavior of traffic we shall consider the *variance of the vehicles speed* in order to determine the traveling time. In this case, the *variance of speed* during a day becomes *a function of time*. This means that the speed S is no longer a function just of i and j but we have $S(i, j, time)$. An example for such a speed graph is displayed in Figure 7.1.

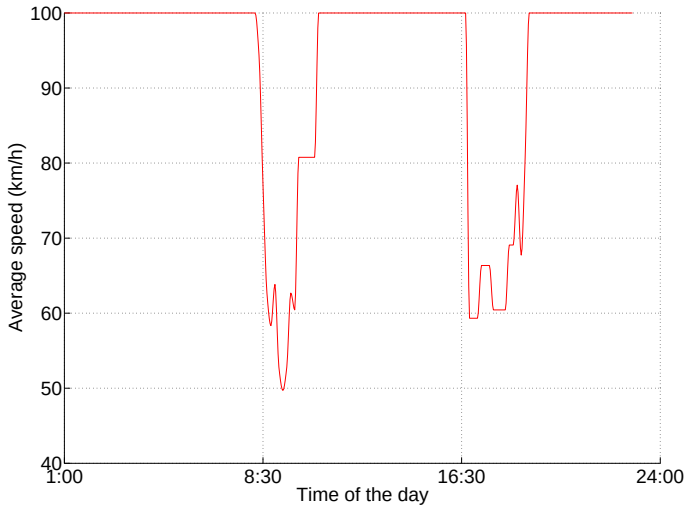
At this step we used data which was created based on our empirical experience. Different speed averages were assigned to highly dense roads for the morning and evening peaks. We used time intervals for each road. The attributed variations of speed fit between 80 and 120 km/h. The traveling time was computed based on these intervals.

7.4 Integrate historical data in the model

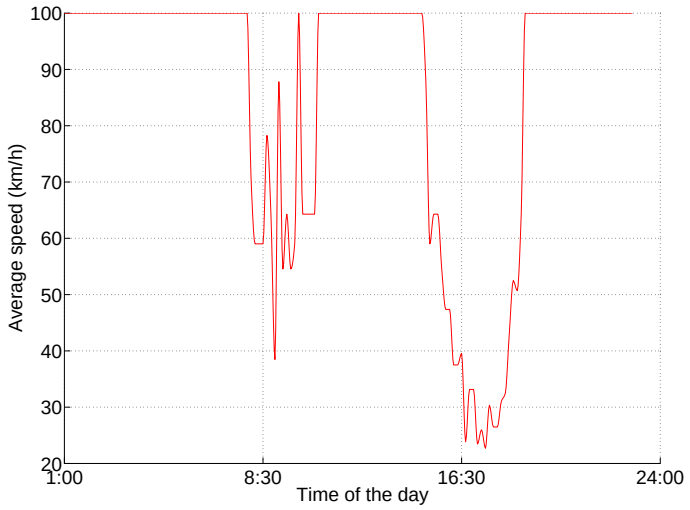
In this phase historical traffic data was integrated in the model. The model uses historical data as the basic data for estimating the traveling time. Recurring traffic patterns such as jams in the traffic network during rush hours can be foreseen by using historical data collected in a similar day. The use of this data allowed a better approximation of the varying speed graphs. We found this an appropriate choice and an important step as it represents a baseline for our research, namely for the dynamic assignment. By using this input data new and different algorithms can be tested and compared.

The historical data is obtained after processing the field data, as explained in Section 4.6.1. Generally, traffic data refers to the average speed or average traveling time of vehicles at different locations in the network. In Section 4.5 the source of the historical data that we use is described. The data from ANWB was collected during 2 months and then analyzed in order to detect analogies and differences between days. There are many factors which can influence the traffic: whether it is a working or a weekend day, the weather conditions, events and incidents. And all these factors influence the traffic assessment as well.

We chose to average the traffic data collected in four Thursdays for all roads in our database. This data allows us to demonstrate the concept of dynamic assignment for a common working day day without any major incidents, events or bad weather. The chosen data was found relevant for a working day. In Figure 7.2 we notice for example that even if there four averaged days the



(a) Knp. DeHoek to Knp. Badhoevedorp (7 km)



(b) Knp. Badhoevedorp to Knp. DeNieuweMeer (3 km)

Figure 7.2: Average of the speed variation during four days

traffic flow has very similar trends for each of them. Thus, at the moment our system estimates the traveling time for a casual working day.

The format of the historical data

First, for each road we computed the traveling time determined by the maximum speed allowed. Next, for each road the the day is divided in 10 minutes intervals. For each interval the extra traveling time obtained from traffic data is added. Missing data was filled in by interpolation using surrounding data. In our system we did not use the fixed time intervals (of 10 minutes) because there were cases in which the interval could be larger. If there is no congestion the traveling time remains constant for longer periods. During night for examples the traveling time was always the same. Hence, consecutive intervals with constant traveling time were added in the database as one interval.

The graphs in Figure 7.2 use the historical data employed also by our algorithm. The graphs show the speed on two highways in The Netherlands close to the ring of Amsterdam. This data was collected on four Thursdays. The graphs use data collected on the two roads at intervals of 10 minutes. In order to represent it in Mat lab we used a *cubic* interpolation.

Figure 7.2a shows that on the road from Knp. DeHoek to Knp. Badhoeverdorp the lowest traffic flow was detected at around 9:12 having around 49 km/h. The maximum allowed speed for both roads is 100 km/h.

Figure 7.2b shows as well that the highest level of congestion between Knp. Badhoevedorp and Knp. DeNieuweMeer is reached at around 17:40 with an average speed of just 22 km/h.

The algorithm uses such data for each road in order to *estimate the traveling time at future states of the network*. It has to return the minimum path in time for a request. In the algorithm the traveling time is calculated dynamically on each road by taking into account the hour at which the vehicle arrives at each intersection. This does not depend on the length of the time intervals associated to each road. More details about how the algorithm implements the dynamics of traffic and calculates the traveling time are presented in the next Section, 7.5.

Further, we can update the historical data by using real traffic information and create traffic models which are presented in the Section 7.5.1.

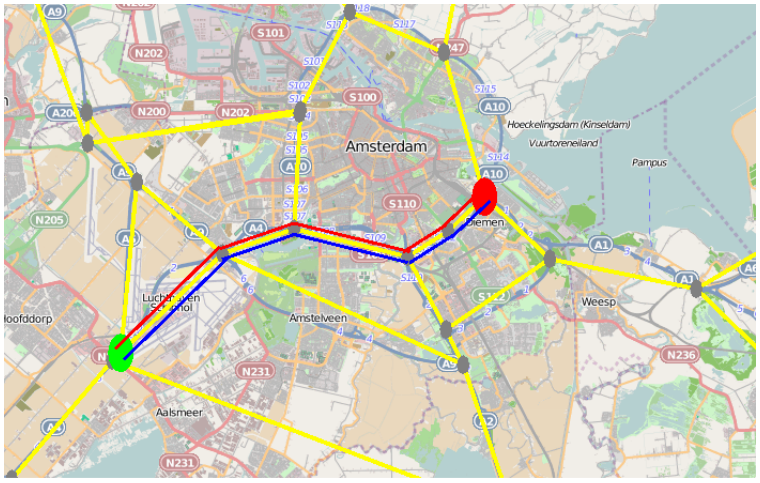


Figure 7.3: Route in the southeast of Amsterdam

7.5 Travel time prediction in the algorithm

In the prototype that we implemented we use a dynamic version of the Dijkstra shortest path. The algorithm is called DDTA (**D**ynamic **D**ijkstra **T**raffic **A**ssignment). The performance of this

algorithm is evaluated in comparison to the static Dijkstra algorithm with regard to the prediction of the traveling time. The algorithm is presented more in detail in Chapter 8.

In this Section we aim to show that the DDTA algorithm selects the speed differently. We will choose a route in the southeast of Amsterdam and exemplify the way both algorithms behave. Figure 7.3 emphasizes the route that we study: starting from Knp. De Hoek to Watergraafsmeer on a distance of almost 21 kilometers. We assume that the route request is done in the afternoon at 17:30. The static algorithm computes the traveling time at that hour by using the maximum speed allowed. The dynamic algorithm checks for each road from the route the traffic flow at the current time. This difference is displayed in Figure 7.4.

Figure 7.4b represents a 3D view of the speed in time along the distance of the route. The graph was plotted in Matlab and it uses the historical data to compute the traveling time for the dynamic approach. We notice that the red line which stands for the dynamic algorithm is more accurate with regard to the estimation of the travel time. We can conclude that the two algorithms use different speeds, dependent on the departure time (the dynamic algorithm) or independent of the departure time (the static algorithm).

7.5.1 Traffic models based on real time updates

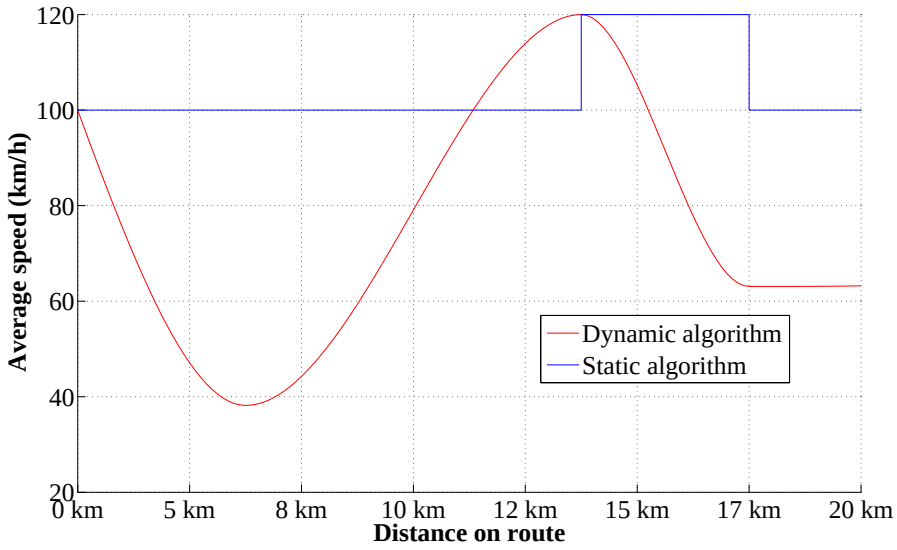
The historical data that our model uses represents an average of 4 Thursdays. These days were found most appropriate for defining the historical data for a casual day of the week. We can notice in the graphs that display the speed graph that the traffic trends were quite similar as we clearly distinguish the peak hours (Figure 7.2). Current traffic updates can be used in order to update the historical data that we already have in order to define other types of days. As we already mentioned in the literature review, there are many factors that influence the distribution of the traffic in the network.

Thus, we consider using live traffic data (such as the average speed of the flow, the number of vehicles that are in the network etc.) obtained from the position reporters of the GPS-equipped mobile phones in order to build various traffic models. These models can represent different days of the week or they can be specific to the weather conditions (like rainy, snowy days etc.). A modality to define this in our model is to keep different intervals for each type.

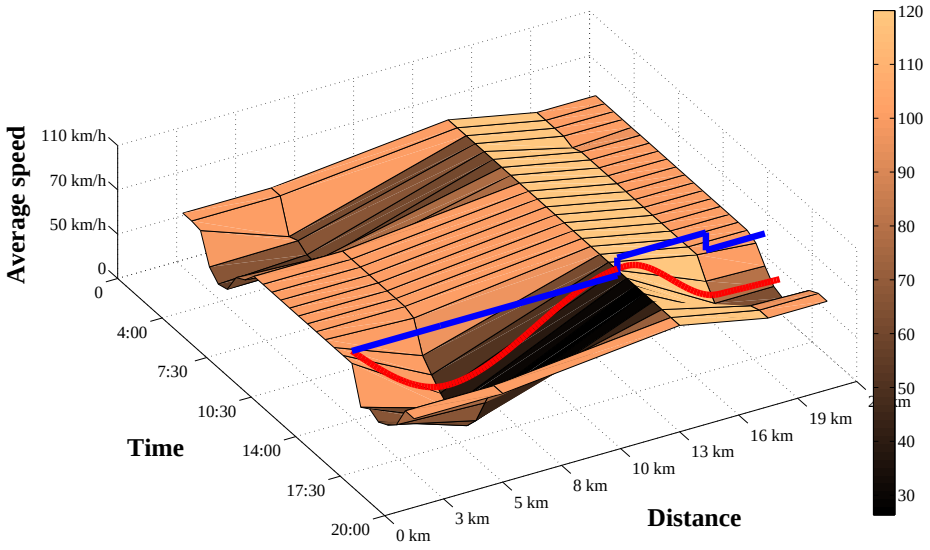
7.5.2 Incidents model

In order to have the possibility to introduce incidents in our model we will develop a simple model as this was not the goal of this thesis. Moreover, due to time constraints we will describe a more advanced approach for an incidents management model in the future work section of this thesis.

However, there are two essential stages for developing an incident management model: we first have to detect incidents and then to manage them. In order to detect an incident we currently use the live traffic data and if the average speed decreases with more than a pre-defined threshold value, ϵ we will increase the traveling time for that road for an interval of one hour. The traveling time increases directly proportional with the traffic stream that was on the road at the moment of the incident. This is because if there were a lot of cars, the delay that is produced is longer. Then, the algorithm has to re-route the vehicles on an alternative path.



(a) Details of the read data. The static algorithm uses just the maximum speed while the dynamic algorithm has a better accuracy



(b) Graph showing the collected data for speed and the data used by the algorithms

Figure 7.4: Differences between static and dynamic algorithm with regard to travel time estimation

Chapter 8

DDTA (Dynamic Dijkstra Traffic Assignment) algorithm

The starting point of the implementation for a dynamic traffic assignment is to build the traffic network as a time expanded graph. Given the traffic model that we presented in the previous chapter we need to implement the algorithm on a graph that is extended in time. This is determined by the time varying speed graphs, which we also presented in the previous chapter. Using this representation we can then apply known mathematical algorithms to solve our problem. As for general routing problems, the Dijkstra's shortest path algorithm could be applied. The main difference to a classic traffic assignment, in representing the network graph is that the cost varies in time.

For our model, the nodes in the graph stand for the intersections in real life but the cost on the links of the graph stand for the traveling time which varies depending on the time of the day.

In order to solve this dynamic assignment a routing algorithm can be applied on the graph propagated in time. Thus, we implemented a dynamic version of the shortest path Dijkstra algorithm that we named DDTA (Dynamic Dijkstra Traffic Assignment).

The inputs to DDTA, the processings and the output of the algorithm are presented in Section 8.1. Along with these we will present the strengths and the weaknesses of our algorithm. A graphical representation of a time expanded graph is presented in Section 8.1.1. The pseudocode, together with a discussion about the differences to the classic Dijkstra algorithm are presented in Section 8.1.2. In that section we also give an example, to show on a synthetic network why a dynamic approach is more appropriate than a static one in the conditions of a time varying flow. We conclude the chapter with results obtained on the real graph that uses real historical data in Section 8.2.

8.1 The algorithm

8.1.1 Inputs to the algorithm

The input to the algorithm is represented by a route request. A route request consists of an O-D (origin-destination) trip demand at a specific time. The algorithm will be applied on the network graph and it uses the nodes, the roads and the estimation of the traveling time based on the varying average speed associated to each road.

The network. The network is represented by a graph $G = \{N, A\}$ where A is the set of directed links and N a set of nodes. G represents the spatial network, meaning the network of nodes and roads.

In order to represent the dynamic travel time we will use a *time extended* network. The time expanded network can be constructed in the following way: the planning horizon is divided into variable time periods ($t = 1, ..T$) and each node is "copied" for each period t so that for each node k there are now T time-space nodes denoted kt. For each link j in the spatial network consider

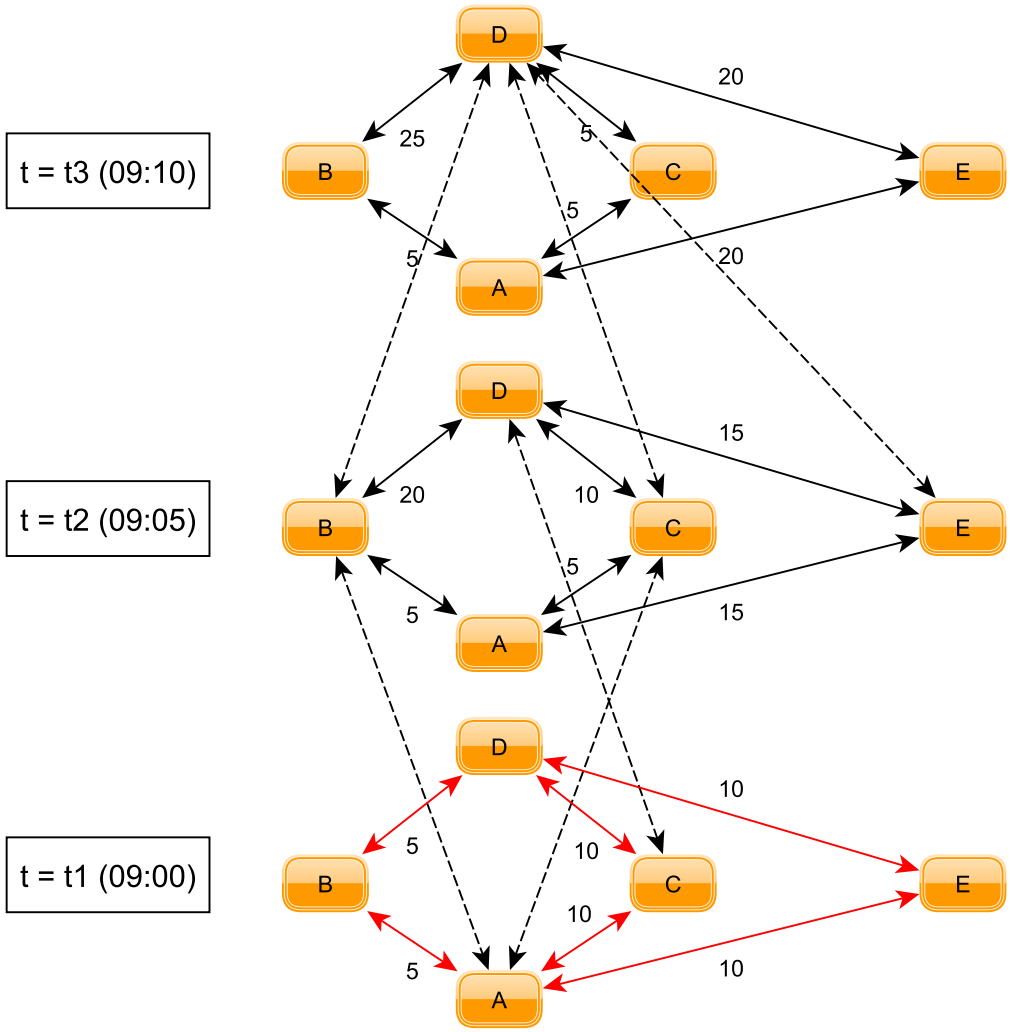


Figure 8.1: Example of a time extended network

time-spaced links, $jt\tau$ joining the entry node of link j at each time t to the exit nodes of link j at latter times, $\tau = t + 1, t + 2, ...$ Thus for each spatial link we have time-expanded links $(jt\tau), \tau = 1, ...T, t = 1, ...T$.

This approach brings one constraint: the travel time has to be discretized to intervals. If we use a very high sample rate then an enormous graph is required whereas a lower sample rate results in loss of information.

An example of a space time extended graph constructed in the modality that we just described is presented in Figure 8.1. The space graph represented by the nodes {A, B, C, D, E} is repeated for three time intervals (t_1 at 09:00, t_2 at 09:05 and t_3 at 09:10). The edges that connect the nodes from A to E, colored in red represent the initial connections in the graph. For clarity these edges were kept similar also for the other layers. But the edges in dotted lines are the real connections of the time expanded graph. They show the evolution in time of the speed flow along with the traveling time in the network. Their length, between the layers, represents the traveling time associated to the corresponding edge when starting at each layer. It should be noticed, however, that not all edges were represented in the figure in order to keep it readable. For example, the traveling time from B to D is 5 minutes at 09:00 and 20 minutes at 09:05.

Algorithm 2 Estimation of the traveling time

```
1.function Double computeTime(double currentTime, Intervals i, Roads r)
2.double crtTime = currentTime;
3.double roadLen = r.getLength();
// How much we need to do from the segment to cover the interval i
4.WHILE (roadLen > 0)
5.   IF ((crtTime + i.getAvgspeed() * roadLen / r.getLength())%1430 > i.getEndtime()) THEN
6.      // We see how much we can cover before the interval finishes
7.      roadLen -= (i.getEndtime() - crtTime) / i.getAvgspeed() * r.getLength();
8.      crtTime = i.getEndtime();
9.      i = db.intervals.findIntervalsEntitiesByRoadAndTime(r.getId(), (crtTime + 1)%1430);
10.ELSE
11.      currentTime = crtTime + i.getAvgspeed() * roadLen / r.getLength();
12.      roadLen = 0;
13.   ENDIF
14.ENDWHILE
15.return currentTime;
```

Algorithm 1 Pseudocode for DDTA

```
1. public void applyDDTA(Graph, Intersection source, Intersection dest, Double departureTime):
2. FOR each intersection in G, i ∈ N: // Initializations
3.   arrivalTime[i] := ∞; // Unknown arrival time from source intersection to i
4.   predecesors[i] := undefined ; // Previous node in optimal path from source
5. ENDFOR
6. arrivalTime[source] := departureTime ;
7. Q := N ; // Q represents the set of unvisited nodes
9. WHILE Q is not empty:
10.   extract a node u from Q for which we have the earliest arrival time so far
11.   IF (u = destination) THEN
12.      break ;
13.   ENDIF
14.   neighbourNodes := getNeighbours(u) ;
15.   FOR each intersection n, n ∈ neighbourIntersections AND n ∉ Q
16.   newArrivalTime := 0;
17.   r= edge between u and n ;
18.   interval := time interval entity corresponding to r at time arrivalTime[u] ;
19.   newArrivalTime := computeTime (Intervals interval, arrivalTime[u], Roads r);
20.   IF (newArrivalTime < arrivalTime[n]) THEN
21.      arrivalTime[n] := newArrivalTime ;
22.      predecesors[n] := u ;
23.   ENDIF
24.   remove u from Q ;
25. ENDWHILE
26. return predecesors[].
```

A more detailed example will be presented in the next section using the same graph in order to illustrate the processings of the algorithm.

8.1.2 Processing of the algorithm

In order to describe the main steps of the DDTA algorithm we will present the pseudocode of the algorithm and illustrate the steps using an example. The pseudocode of the algorithm is presented at Figure 1. The algorithm uses a function computeTime() that is presented in Figure 2. The function estimates the traveling time that is necessary in order to traverse the road which is received as parameter.

An example of a time extended graph was illustrated in the previous section at Figure 8.1. We will use the same graph in order to demonstrate an example of how the algorithm processes the input data and returns a route. Let us assume that we have a route request from A to D at 09:00.

The solution for the static routing algorithm that does not include the variation over time is the path A-B-D as it has the smallest cost at 9:00.

The first step represents the initialization of the traveling time ($t[]$) and the predecesors ($p[]$) vectors. The set of unvisited nodes gets initialized with the set of nodes, Q = {A, B, C, D, E} and the $t[] = [09:00, \infty, \infty, \infty, \infty]$. At the first step we extract A out of Q as it has the smallest time. The neighbors of A are B, C, E for which the arrival time is computed as 09:05 for B, 9:10 for C and 9:10 also for E as the traveling times on these edges at 09:00 is 5, 10 and again 10. The values in the travelTime vector change only if the new arrival times are earlier. Thus, $t[]$ =[09:00, 9:05, 9:10, ∞, 9:10] and $p[]$ =[undefined, A, A, undefined, A].

At next step, B is extracted from Q as it has the earliest arrival time from Q. Its neighbors are {A,D} and the traveling time is computed just for D as A has already been visited. So, the arrival time to D through B would be 9:25, as we add 20 minutes to 9:05. Here we already consider the second layer of the time extended graph.

Next, C will be extracted from Q and its neighbors will be {A, D}. As both have already been visited we extract the next node from Q with the earliest arrival time, that is E. Again, his neighbors are {A, D} which have been already visited. The last node that remains in Q is D which happens to be the destination. At this point we have $t[]$ = [9:00, 9:05, 9:10, 9:15, 9:10] and $p[]$ =[undefined, A, A, C, A]. In order to see which is the path we use $p[]$ where each element represents the predecessor of the node corresponding to the position in the vector: $p[D] = C$ and $p[C] = A$ which is the source. The car would reach its destination at 09:15.

As we can notice, the main difference between DDTA algorithm and the static version, is that DDTA knows the arrival time at each node and computes the traveling time on the next connection corresponding to this.

8.1.3 Outputs of the algorithm

Algorithm 3 Composition of the route solution

```
1. public Route getDDTASolution(Node source, Node destination, double departureTime):
2. S := routeSolution ;
3. u := destination ;
4. predecesors[] := applyDDTA(source, target, departureTime) ;
5. WHILE predecesors[u] is defined
6.     insert u at the beginning of S ;
7.     u := predecesors[u] ;
8. ENDWHILE
9. return S.
```

DDTA is applied on the network graph along with the traveling time which varies during the day. The result of this is mainly represented by a vector of predecessor nodes which are used to rebuild the path. This is shown in the Pseudocode 3. Another output of the algorithm is represented by the arrival time at the destination.

The output of the algorithm is stored in a new object that represents a route solution. We need this object in order to do some further processings which are needed for the information that we want to show in the interface to the user. Here we show which are the time differences in compare to the static algorithm. These differences refer to the estimated traveling time of both algorithms and the traveling time computed using the historical data (which is more realistic). Other procedures which are implemented refer to the computation of the fuel consumption for the route or the length of the route.

8.1.4 Strengths and Weaknesses

When discussing strength and weaknesses we will focus on the most important ones, considering our use cases. Given the traffic in The Netherlands is very dynamic as the conditions change fast and unexpected events occur one of the biggest advantages of DDTA is that it can adapt fast to the new situations. This is mainly due to the fact that the traffic model relies on collected data

and it is not fixed. The only requirement is that the data is updated in real time from one of the sources described in previous chapters.

One disadvantage of DDTA is that the computation can take long if the number of nodes will increase to include also less important roads. This could be fixed by extending it with an heuristic that would make the search a guided search.

8.2 Results

In order to illustrate the differences between the output of DDTA and a classic routing algorithm we will choose some routes between important cities in The Netherlands and we compute travel time between them. We do this for various times of day, to illustrate that our algorithm performs better during rush hours.

We test the algorithms on routes that start and end on the rings of the major cities like Amsterdam, Rotterdam, The Hague, Utrecht, Eindhoven, Nijmegen, or Groningen. We will group these routes by the pair of cities from which they start/end such as Amsterdam-Rotterdam, The Hague-Eindhoven etc. These cities were chosen so that the resulting routes cover a large part of the map. The roads used by the algorithm are mostly highways and a few national roads from The Netherlands.

We will initially present the results of our algorithm compared to the Static Dijkstra algorithm (SDA). SDA uses as a cost metric the distance between intersections. Thus, this algorithm uses a completely different assessment of the performance measure. That is why we also implemented a version of the Dijkstra algorithm that has as a cost measure traveling time. The difference to DDTA is that this algorithm checks and computes the shortest traveling time by using the maximum speed on the roads. In other words, it does not check the average speeds during the trip, in time. We will call this algorithm Improved Dijkstra Algorithm (IDA).

The steps that we took in order to generate the results are the following:

- Select intersections around the major cities. For example, in case of Amsterdam we have 7 intersections: Knooppunt Coenplein, Knooppunt De Nieuwe Meer, Knooppunt Amstel, Knooppunt Watergraafsmeer and the intersections between {A10, N247}, {A10, N200}, {A10, S112}. Next, we generate all possible routes between each pair of cities (such as Amsterdam-Rotterdam).
- Apply DDTA for the roads between each 2 cities at each 30 minutes during the whole day. The other 2 algorithms (STA and IDA) are also applied.
- Compute an *average traveling time* between routes at the same hours for each pair of cities using the result of DDTA.
- Compute an *average traveling time* in the same conditions after applying the STA.
- Compute the *average difference* in traveling times associated to DDTA and the ones to STA.
- Show the *maximum difference* that appears between the algorithms, when applied on the routes between a pair of cities.

Average traveling time

In order to show the differences in the traveling time we chose to apply all three algorithms (DDTA, STA and IDA) for the same routes. We selected the intersections around the cities and connected them. In this way, we obtained a couple of routes linking 2 cities. The algorithms are applied at each 30 minutes during the day but we choose to illustrate the results just for the peak hours. In the night or in the evening when there are no congestions the results are usually similar.

Thus, we averaged the traveling times that belong to the routes between 2 cities. For the morning peak we chose to request the routes at 06:30, 07:00, 07:30, 08:00, 08:30, 09:00 and 09:30. These results are illustrated in Table 8.1. In order to be more clear we will explain the differences

Table 8.1: Average traveling times for DDTA, SDA and IDA (at morning peak)

DDTA								
		06:30	*07:00*	*07:30*	*08:00*	*08:30*	*09:00*	*09:30*
Amsterdam	Rotterdam	47,72	47,74	50,33	55,23	57,11	57,24	57,88
Rotterdam	Amsterdam	50,08	56,59	60,56	62,97	64,91	62,79	58,11
Eindhoven	Den Haag	79,59	81,61	83,3	83,22	81,72	79,84	77,96
Den Haag	Eindhoven	79,59	81,61	83,3	83,22	81,72	79,84	77,96
Eindhoven	Amsterdam	76,31	81,19	84,76	86,99	84,68	82,87	79,67
Amsterdam	Eindhoven	67,33	67,99	76,76	82,26	85,47	82	81,35
Utrecht	Amsterdam	23,91	26,69	27,14	32,92	33,53	29,93	26,27
Amsterdam	Utrecht	23,72	23,76	24,28	28,61	35,52	36,12	35,82
Amsterdam	Nijmegen	63,25	67,47	69,53	71,21	77,25	74,86	67,95
Nijmegen	Amsterdam	69,26	67,41	76,81	81,22	72,77	71,69	67,09
SDA								
		06:30	*07:00*	*07:30*	*08:00*	*08:30*	*09:00*	*09:30*
Amsterdam	Rotterdam	51,11	51,12	53,95	57,41	60,01	63,81	62,66
Rotterdam	Amsterdam	55,67	59,75	62,34	73	72,75	66,44	61,95
Eindhoven	Den Haag	84,28	84,9	85,17	85,93	85,75	84,64	83,47
Den Haag	Eindhoven	84,28	84,9	85,17	85,93	85,75	84,64	83,47
Eindhoven	Amsterdam	77,08	82	85,65	88,15	85,77	84,21	81,41
Amsterdam	Eindhoven	68,15	68,95	77,34	82,97	86,02	83,02	82,56
Utrecht	Amsterdam	24,75	27,21	28,38	34,17	34,18	31,88	27,41
Amsterdam	Utrecht	24,73	24,73	24,81	29	35,87	36,85	37,64
Amsterdam	Nijmegen	63,25	69,22	72,2	71,72	78,8	74,96	71,19
Nijmegen	Amsterdam	70,11	70,61	77,68	81,93	73,83	71,69	68,45
IDA								
		06:30	*07:00*	*07:30*	*08:00*	*08:30*	*09:00*	*09:30*
Amsterdam	Rotterdam	47,97	47,99	50,69	55,4	59,04	61,88	59,75
Rotterdam	Amsterdam	50,57	57,07	67,06	71,94	72,23	65,63	60,45
Eindhoven	Den Haag	79,72	83,72	94,27	97,61	93,83	84,7	82,32
Den Haag	Eindhoven	79,72	83,72	94,27	97,61	93,83	84,7	82,32
Eindhoven	Amsterdam	76,57	81,44	85,02	87,33	84,94	83,29	80,4
Amsterdam	Eindhoven	67,59	68,3	77,26	82,46	85,47	82	81,9
Utrecht	Amsterdam	24,5	26,95	28,13	33,87	33,79	31,44	26,53
Amsterdam	Utrecht	24,47	24,47	24,54	28,8	35,52	36,12	37,25
Amsterdam	Nijmegen	63,25	69,22	72,2	71,72	78,8	74,96	71,19
Nijmegen	Amsterdam	70,11	70,61	77,68	81,93	73,83	71,69	68,45

for the 3 algorithms on one example. Let us take the routes between Amsterdam and Rotterdam. At 09:00, for the same routes the average traveling time given by DDTA is 57,24 minutes while the STA gives an average of 63,81 minutes.

Table 8.2 illustrates the same results but for the afternoon time.

Average difference of traveling time and maximum differences

In order to show more clearly which are the differences between the DDTA and STA averages of the traveling time we use Table 8.3. In this table the first part shows the average time that DDTA gains compared to STA in minutes. Some values here may appear to be lower than expected but these are the averages. It is worth mentioning that in the worst case DDTA gives a result with the same traveling time as the other algorithms, but never worse. This is the case of the 0 values in the second part of the table. In this part we show at each hour the route with greatest gain

Table 8.2: Average traveling times for DDTA, SDA and IDA (at afternoon peak)

DDTA														
		13:00	*13:30*	*14:00*	*14:30*	*15:00*	*15:30*	*16:00*	*16:30*	*17:00*	*17:30*	*18:00*	*18:30*	*19:00*
Amsterdam	Rotterdam	47,64	47,64	47,64	47,94	50,54	56,7	62,87	66,5	73,92	74,68	68,19	60,68	56,51
Rotterdam	Amsterdam	47,64	48,48	49,22	48,76	48,65	52,66	56,57	62,82	64,91	63,62	62,64	57,96	52,61
Eindhoven	Den Haag	71,89	71,79	71,79	71,79	71,81	78,6	81,19	81,95	86,54	85,55	84,62	81,07	75,96
Den Haag	Eindhoven	71,89	71,79	71,79	71,79	71,81	78,6	81,19	81,95	86,54	85,55	84,62	81,07	75,96
Eindhoven	Amsterdam	67,43	67,33	67,34	67,4	68,99	72,8	73,71	80,31	85,51	88,63	88,63	76,02	68,03
Amsterdam	Eindhoven	67,33	67,33	67,33	68,4	74,36	96,99	112,58	118,51	115,42	105,11	102,2	92,09	83,81
Utrecht	Amsterdam	23,72	23,72	23,72	23,84	23,73	24,27	27,58	29,96	29,68	29,92	31,7	28,67	29,89
Amsterdam	Utrecht	23,72	23,72	23,72	23,75	25,78	32,7	40,01	42,39	44,33	36,88	37,1	34,58	32,97
Amsterdam	Nijmegen	63,25	63,25	63,25	63,62	68,94	85,47	100,66	104,65	103,78	95,22	95,53	82,43	77,54
Nijmegen	Amsterdam	63,25	63,25	63,25	63,34	63,57	67,68	67,73	70,48	72,66	76,55	79,21	69,38	64,76
SDA														
		13:00	*13:30*	*14:00*	*14:30*	*15:00*	*15:30*	*16:00*	*16:30*	*17:00*	*17:30*	*18:00*	*18:30*	*19:00*
Amsterdam	Rotterdam	51,11	51,11	51,11	52,04	56,19	61,99	68,66	76,11	82,66	82,87	76,14	66,33	59,91
Rotterdam	Amsterdam	51,33	51,76	52,16	53,39	54,74	59,63	65,13	73,35	73,51	69,88	67,07	62,6	54,78
Eindhoven	Den Haag	83,47	83,47	83,47	83,47	83,63	85,22	85,04	86,13	89,61	89,2	89,38	86,28	83,47
Den Haag	Eindhoven	83,47	83,47	83,47	83,47	83,63	85,22	85,04	86,13	89,61	89,2	89,38	86,28	83,47
Eindhoven	Amsterdam	68,53	68,47	68,47	68,88	70,88	75,08	78,6	84,3	88,56	90,57	90,4	76,78	68,93
Amsterdam	Eindhoven	68,15	68,15	68,18	69,32	75,67	98,13	116,09	121,91	121,91	109,87	102,63	92,5	84,23
Utrecht	Amsterdam	24,73	24,93	25,05	25,19	25,26	26,28	30,02	34,64	32,18	33,93	34,65	29,84	30,35
Amsterdam	Utrecht	24,73	24,73	24,73	24,8	26,59	34,51	41,35	45,44	46,53	38,35	37,68	34,72	33,15
Amsterdam	Nijmegen	63,25	63,25	63,25	63,62	69,24	85,5	107,97	108,52	111,82	101,43	96,13	83,1	77,54
Nijmegen	Amsterdam	63,25	63,25	63,25	63,34	63,94	67,91	69,14	70,96	78,93	82,68	83,23	69,59	64,76
IDA														
		13:00	*13:30*	*14:00*	*14:30*	*15:00*	*15:30*	*16:00*	*16:30*	*17:00*	*17:30*	*18:00*	*18:30*	*19:00*
Amsterdam	Rotterdam	47,9	47,9	47,9	48,26	50,84	58,96	70,28	77,58	84,17	80,35	73,33	64,54	58,96
Rotterdam	Amsterdam	47,95	48,77	49,85	49,45	49,49	55,13	62,55	68,72	73,81	67,1	63,6	60,56	55,77
Eindhoven	Den Haag	71,89	71,79	71,79	71,79	71,81	80,05	85,05	98,81	101,1	103,69	99,29	87,11	76,36
Den Haag	Eindhoven	71,89	71,79	71,79	71,79	71,81	80,05	85,05	98,81	101,1	103,69	99,29	87,11	76,36
Eindhoven	Amsterdam	67,68	67,59	67,59	67,66	69,01	73,00	74,66	81,26	86,55	90,21	91,24	76,88	68,29
Amsterdam	Eindhoven	67,59	67,59	67,6	68,7	74,44	97,36	115,22	120,72	121,4	109,04	102,22	92,09	83,81
Utrecht	Amsterdam	24,47	24,47	24,47	24,59	24,48	24,9	28,5	32,3	30,18	30,84	32,3	28,71	29,92
Amsterdam	Utrecht	24,47	24,47	24,47	24,52	26,06	33,94	40,78	44,64	46,31	37,68	37,55	34,58	32,97
Amsterdam	Nijmegen	63,25	63,25	63,25	63,62	69,24	85,5	107,97	108,52	111,82	101,43	96,13	83,1	77,54
Nijmegen	Amsterdam	63,25	63,25	63,25	63,34	63,94	67,91	69,14	70,96	78,93	82,68	83,23	69,59	64,76

in time for DDTA. We notice for example that from Rotterdam to Amsterdam we have a gain of 16,78 minutes at 8 o'clock.

The last part of Table 8.3 shows the routes for which DDTA had the greatest gain in time, compared to IDA. As we already mentioned IDA searches for the shortest traveling time at departure. Here we notice a difference of almost 25 minutes from Rotterdam to Amsterdam at 8 o'clock.

The same results for the afternoon peak hours are presented in Table 8.4.

Table 8.3: DDTA compared to SDA and IDA - Maximum differences (morning peak hours)

Average travel times difference (DDTA to SDA)								
		06:30	*07:00*	*07:30*	*08:00*	*08:30*	*09:00*	*09:30*
Amsterdam	Rotterdam	3,39	3,38	3,62	2,19	2,9	**6,57**	4,78
Rotterdam	Amsterdam	5,6	3,16	1,78	**10,04**	7,84	3,65	3,84
Eindhoven	Den Haag	4,7	3,29	1,87	2,7	4,03	4,8	**5,51**
Den Haag	Eindhoven	4,7	3,29	1,87	2,7	4,03	4,8	**5,51**
Eindhoven	Amsterdam	0,78	0,81	0,9	1,16	1,09	1,34	**1,75**
Amsterdam	Eindhoven	0,82	0,96	0,58	0,71	0,55	1,02	**1,22**
Utrecht	Amsterdam	0,84	0,51	1,25	1,25	0,65	**1,96**	1,14
Amsterdam	Utrecht	1,01	0,96	0,53	0,38	0,34	0,72	**1,82**
Amsterdam	Nijmegen	0	1,74	2,67	0,51	1,55	0,09	**3,24**
Nijmegen	Amsterdam	0,85	3,21	0,86	0,72	1,07	0	**1,36**
Maximum travel time difference when comparing DDTA to SDA								
		06:30	*07:00*	*07:30*	*08:00*	*08:30*	*09:00*	*09:30*
Amsterdam	Rotterdam	**12,55**	12,47	12,84	8,23	6,36	11,14	11,96
Rotterdam	Amsterdam	15,83	11,74	8,42	**16,73**	13,26	9,41	13,87
Eindhoven	Den Haag	**7,18**	6,22	4,6	2,85	4,86	5,76	5,51
Den Haag	Eindhoven	**7,18**	6,22	4,6	2,85	4,86	5,76	5,51
Eindhoven	Amsterdam	3,97	4,09	4,71	6,00	7,15	9,99	**10,96**
Amsterdam	Eindhoven	4,25	5,09	2,86	3,71	3,57	**6,8**	5,55
Utrecht	Amsterdam	4,53	2,4	5,97	6,05	3,56	**8,77**	7,15
Amsterdam	Utrecht	5,00	4,75	2,49	1,86	2,41	**5,06**	7,5
Amsterdam	Nijmegen	0	5,3	7,46	2,38	4,06	0,46	**9,42**
Nijmegen	Amsterdam	2,28	**8,12**	4,05	3,36	5,21	0,07	7,06
Maximum travel time difference when comparing DDTA to IDA								
		06:30	*07:00*	*07:30*	*08:00*	*08:30*	*09:00*	*09:30*
Amsterdam	Rotterdam	1,08	0,9	0,94	0,6	4,13	**9,44**	4,12
Rotterdam	Amsterdam	3,05	3,59	16,93	**24,85**	18,59	12,9	12,94
Eindhoven	Den Haag	1,56	5,63	15,56	**18,67**	17,97	8,58	7,95
Den Haag	Eindhoven	1,56	5,63	15,56	**18,67**	17,97	8,58	7,95
Eindhoven	Amsterdam	0,6	0,6	0,6	1,14	0,6	1,52	**2,96**
Amsterdam	Eindhoven	0,6	0,9	**2,25**	0,85	0	0	1,38
Utrecht	Amsterdam	2,73	0,6	4,17	4	0,6	**5,38**	0,6
Amsterdam	Utrecht	3,2	3,12	0,62	0,47	0	0	**4,77**
Amsterdam	Nijmegen	0	5,3	7,46	2,38	4,06	0,46	**9,42**
Nijmegen	Amsterdam	2,28	**8,12**	4,05	3,36	5,21	0,07	7,06

Table 8.4: DDTA compared to SDA and IDA - Maximum differences (afternoon peak hours)

Average travel times difference (DDTA to SDA)														
		13:00	*13:30*	*14:00*	*14:30*	*15:00*	*15:30*	*16:00*	*16:30*	*17:00*	*17:30*	*18:00*	*18:30*	*19:00*
Amsterdam	Rotterdam	3,47	3,47	3,47	4,1	5,65	5,29	5,79	**9,61**	8,75	8,2	7,95	5,65	3,41
Rotterdam	Amsterdam	3,69	3,28	2,94	4,63	6,09	6,96	8,56	**10,53**	8,6	6,26	4,43	4,64	2,16
Eindhoven	Den Haag	11,58	11,68	**11,68**	11,68	11,82	6,62	3,85	4,19	3,07	3,65	4,76	5,21	7,51
Den Haag	Eindhoven	11,58	11,68	**11,68**	11,68	11,82	6,62	3,85	4,19	3,07	3,65	4,76	5,21	7,51
Eindhoven	Amsterdam	1,11	1,14	1,14	1,47	1,89	2,28	**4,89**	3,99	3,05	1,94	1,77	0,76	0,9
Amsterdam	Eindhoven	0,82	0,82	0,84	0,93	1,31	1,14	3,51	3,4	6,5	**4,76**	0,43	0,41	0,42
Utrecht	Amsterdam	1,01	1,2	1,33	1,35	1,53	2,01	2,44	**4,68**	2,5	4,01	2,95	1,18	0,46
Amsterdam	Utrecht	1,01	1,01	1,01	1,05	0,81	1,8	1,34	**3,04**	2,2	1,47	0,59	0,14	0,18
Amsterdam	Nijmegen	0	0	0	0	0,3	0,03	7,31	**3,87**	8,05	6,2	0,6	0,67	0
Nijmegen	Amsterdam	0	0	0	0	0,37	0,23	1,41	0,48	**6,27**	6,13	4,02	0,21	0
Maximum travel time difference when comparing DDTA to SDA														
		13:00	*13:30*	*14:00*	*14:30*	*15:00*	*15:30*	*16:00*	*16:30*	*17:00*	*17:30*	*18:00*	*18:30*	*19:00*
Amsterdam	Rotterdam	12,55	12,55	12,55	14,06	15,87	12,95	11,79	17,62	**21,51**	15,62	17,63	13,63	8,13
Rotterdam	Amsterdam	12,55	11,36	10,05	14,15	18,54	16,75	20,52	**21,82**	21,94	13,74	10,42	10,69	9,26
Eindhoven	Den Haag	13,91	13,91	13,91	13,91	**13,91**	9,35	4,85	4,49	3,75	5,09	5,43	5,55	9,19
Den Haag	Eindhoven	13,91	13,91	13,91	13,91	**13,91**	9,35	4,85	4,49	3,75	5,09	5,43	5,55	9,19
Eindhoven	Amsterdam	6,37	6,5	6,5	8,92	14,02	14,32	**26,99**	22,95	17,98	9,8	6,28	4,01	4,77
Amsterdam	Eindhoven	4,25	4,25	4,38	4,79	8,51	7,12	7,74	9,93	**10,96**	8,75	2,55	2,59	2,61
Utrecht	Amsterdam	5	6,34	7,25	7,57	9,11	13,78	13,88	**24,75**	21,5	22,89	17,14	10,68	3,3
Amsterdam	Utrecht	5	5	5	5,33	5,07	9,81	5,44	10,46	8,27	**10,67**	3,78	1	1,63
Amsterdam	Nijmegen	0	0	0	0	2,24	0,64	**12,21**	7,32	10,64	11,07	2,58	2,56	0
Nijmegen	Amsterdam	0	0	0	0	2,87	1,94	9,99	2,59	12,27	**13,28**	10,6	1,47	0
Maximum travel time difference when comparing DDTA to IDA														
		13:00	*13:30*	*14:00*	*14:30*	*15:00*	*15:30*	*16:00*	*16:30*	*17:00*	*17:30*	*18:00*	*18:30*	*19:00*
Amsterdam	Rotterdam	1,15	1,15	1,15	2,52	3,76	10,97	15,14	**32,1**	28,84	18,04	13,7	12,55	9,22
Rotterdam	Amsterdam	2,65	1,05	3,03	5,77	9,31	10,81	16,64	17,68	**21,32**	14,08	4,04	9,83	6,4
Eindhoven	Den Haag	0	0	0	0	0	7,04	11,92	**25,1**	20,53	20,46	20,62	10,25	3,34
Den Haag	Eindhoven	0	0	0	0	0	7,04	11,92	**25,1**	20,53	20,46	20,62	10,25	3,34
Eindhoven	Amsterdam	0,6	0,6	0,6	0,6	0,69	2,61	**8,64**	7,63	6,98	4,64	7,98	3,02	0,6
Amsterdam	Eindhoven	0,6	0,6	0,63	0,74	0,21	1,41	4,13	**6,98**	9,11	4,88	0,09	0	0
Utrecht	Amsterdam	3,2	3,2	3,2	3,2	3,2	3,27	3,25	**9,8**	3,85	7,59	4,45	1,6	0,6
Amsterdam	Utrecht	3,2	3,2	3,2	3,39	1,71	5,83	2,67	**10,21**	8,27	4,52	3,78	0	0
Amsterdam	Nijmegen	0	0	0	0	2,24	0,64	12,21	7,32	10,64	**11,07**	2,58	2,56	0
Nijmegen	Amsterdam	0	0	0	0	2,87	1,94	9,99	2,59	12,27	**13,28**	10,6	1,47	0

Chapter 9

Implementation

In this chapter the implementation of PTA will be discussed. A detailed description of the way we implemented and connected the components of the system that we presented at the conceptual design of PTA will be presented. We aim to highlight the most interesting aspects of what we implemented.

Section 9.1 presents which parts from the conceptual design of the system were implemented in our prototype following that the key components of the system to be presented in Section 9.2. Here we introduce a broad description of the system's architecture and environment. We also describe the interaction modality between the components.

Section 9.3 emphasizes the description of the main data structures that were used in the implementation. We stress the description of the objects used in the OO model associated to each data structure. Moving on, we present more details about the database object relational mapping in Section 9.4.

A short description of the client-server communication protocol that we use is presented in Section 9.5. Section 9.6 describes the implementation and the layout of the GUI for the desktop application. It mainly emphasizes the basic functionalities such as the map editing, historical data overview and routing. The routing interface will be given more attention as it represents the main part of the GUI from the desktop application. Thus, Section 9.7 explains how the OSM API was used in order to have an embedded OSM map in the system. It mostly describes the procedure that is used in order to render the map in the GUI.
Finally, we present some details on the map tiles rendering for the GUI of the mobile phone and we discuss how we store the data on the mobile in Section 9.8.

9.1 Prototype

Based on the design of PTA that we presented in Chapter 6 we implemented a prototype. The prototype develops just certain parts of the design that we described. Due to time constraints and complexity of such a system we chose to implement the most important components in order to obtain a feasible working prototype.

So, we focus to implement all the most important parts from the design we presented in Figure 6.8. By comparing this figure to Figure 9.1 (that shows the architecture of the implemented application) we notice which are the differences or in other words, how far we succeeded to implement. No need to mention, there were certain aspects that were beyond the scope of this thesis. In the remaining of this section we will briefly present the main parts that we implemented.

Thus, the prototype is based on a client-server communication. That is why we implemented a central server and two clients which are the desktop application and the mobile phone. The user can use any of these clients in order to ask for a route request and receive an answer.

The server implements the dynamic traffic algorithm that uses a prediction model based on historical data. The prediction model that we implemented is presented in Chapter 7 whereas the

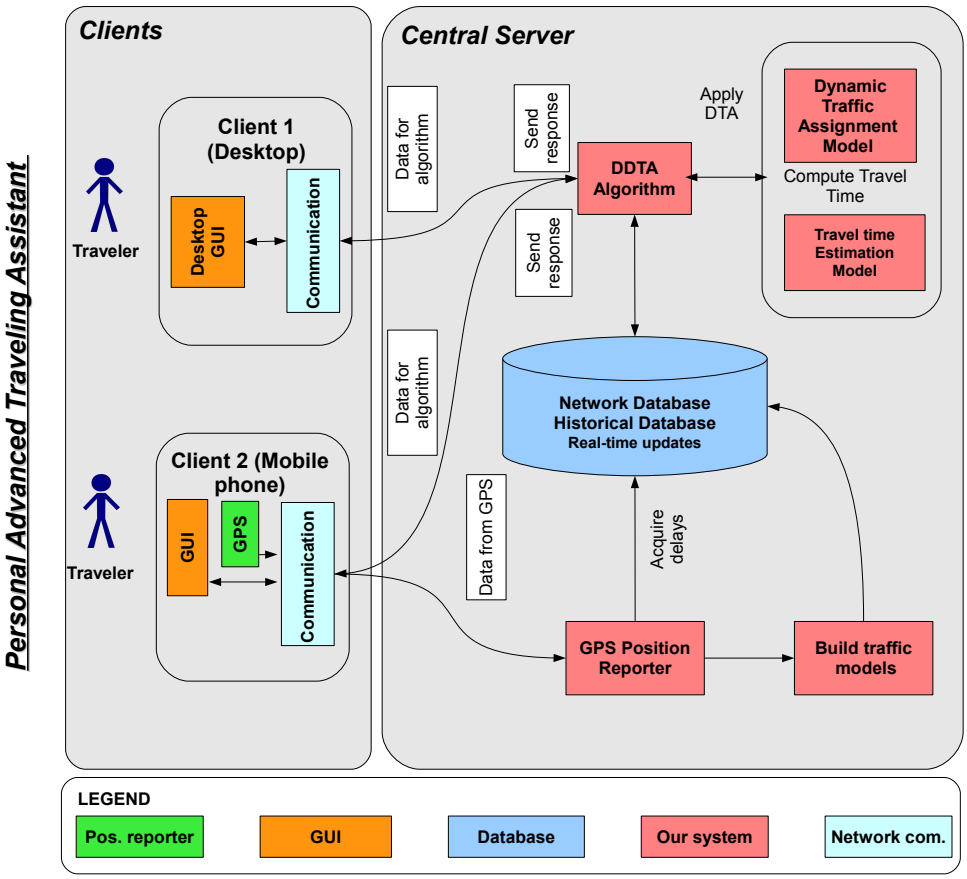


Figure 9.1: Implemented architecture

implemented algorithm is discussed in Chapter 8. The algorithm was tested on real traffic data collected from the highways in The Netherlands.

The prototype uses now only historical data but it can be easily extended to include the live updates. These can be collected from the GPS-equipped mobile phones which regularly send their position. The prediction model can be extended to perform future predictions by combining the current updates with the knowledge about the routes of the guided cars. These would be the most important parameters in order to determine a relation between the traffic flow and future traveling time, train a classifier like an ANN and obtain an accurate future traffic prediction.

With regard to the interfaces of PTA to the users, which are the mobile phone and the desktop application we focused our efforts into developing a graphical user interface (GUI). Even if we mentioned that an ideal PTA would integrate input modalities like the speech recognition, this was considered far beyond the scope of this thesis.

In order to develop the GUI of both clients we closely followed the mock-ups that we presented in Chapter 6.

9.2 System architecture

In this section we will introduce the architecture of the prototype and discuss the interaction between the components. The choice of the components that we implemented was determined by

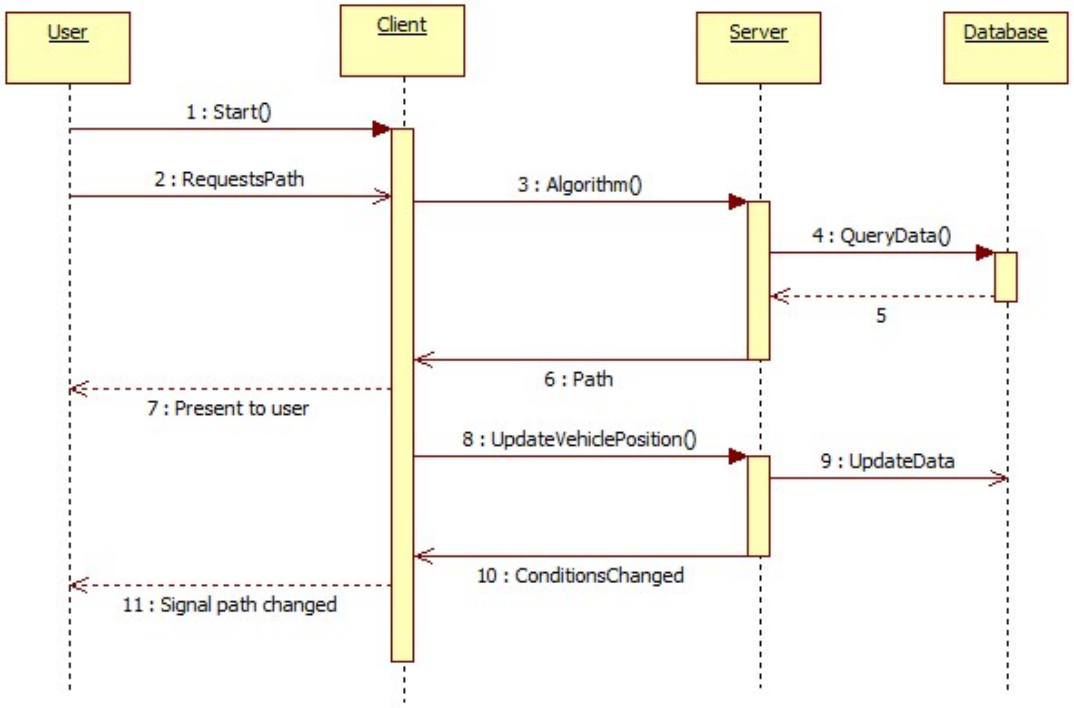


Figure 9.2: Sequence diagram

the environment of PTA presented in Chapter 6. Figure 9.1 shows actually the parts which were implemented. Thus, the system uses a client-server communication whose main components are the following:

- *Central server*: the clients send route requests to the server by using the GUI of the desktop application or the mobile phone. The parameters of the request like the origin, the destination of the trip and the departure time are sent through a TCP/IP protocol to the server. On the server the algorithm is applied on the input. This employs the traffic prediction model that is based on real historical data collected from the highways in The Netherlands. In order to retrieve all the necessary information, such as the nodes, the roads or the time intervals (associated to each road) the algorithm connects to the database. When the solution is computed, it is sent to the client who requested it.
- *Desktop Client*: communicates a route request to the server by giving the origin, the destination of the trip and the departure time. The user asks for a route request by using the GUI of the application. This client has also got graphical editing possibilities for the map. But the access to modify the map would be restricted to the authorized users.
- *Mobile Client*: communicates his destination by clicking on the map or by entering the name. The departure time will be considered the current time. The server receives the request, computes the solution and sends back a route.

In order to present the communication flow in the system we choose to use an UML sequence diagram. Sequence diagrams are usually useful to illustrate the interactions because it is easy to see the call-flow sequence by reading it from top to bottom. Thus, the UML sequence diagram depicted in Figure 9.2 gives more insight into the dynamics of our system. We would like to mention that the flow of messages and actions is highly synthesized in order to show just the main relations between the components.

The vertical time lines show the sequence of intersections between the header elements: the user, the GPS receiver, the GUI, the central server and the database. The user sends a route request by using one of the clients. This calls the algorithm on the server which connects to the database. Next, the server communicates the path and the user is supervised in order to gather current information from the traffic. In case of unexpected events, such as an incident (if conditions change) the server re-routes the client.

9.3 Data structures

The main data structures that are used by the algorithm are: the network graph, the time intervals associated to each road, the cars and the routes. For all these classes the fields are private and we provided accessor methods for each.

Network: intersections and roads. The network is represented by a directed graph. We will call the nodes of the graph intersections and create a Java class named Intersections that will store the attributes of the nodes. For the edges we create a Java class named Roads.

The Intersections class contains the following fields:

- id: Integer. This field is needed as we have to serialize these information to a database, and Integers are very good as primary keys.
- name: String. To be able to manage more easily the nodes, a human readable identification is needed, and we stored this identification in the Name field.
- lat, long1: Double. These are the main fields, as they encode the position of the intersection on the map. They will be used for graphical representation mostly, as we can't rely on them for length computation. The length of the roads between intersections may vary a lot from the straight line as in real life highways take turns. Note: We didn't named longitude just 'long' as this would interfere with the 'long' keyword which describes a Java primitive type.

All the intersections that we use were initially contained in an Excel file. Each node was provided with an associated name. For each location the coordinates had to be separately extracted from the Open Street Map server. Next, each of these nodes was inserted in the database.

The class Road has the following fields:

- fromi: Integer. The id of the intersection from which this road 'leaves'.
- toi: Integer. The id of the intersection to which this road 'arrives'.
- name: String. The name given by the authorities to this road. There will be multiple roads with the same name.
- llength: Double. The length of the road taken from an external database (as we already discussed we can't use the coordinates of the nodes to compute this length). Note: The additional 'l' is not a spelling mistake. The reason is linked to the fact that 'length' is a SQL keyword. More explanations will be given in next section.
- speedlimit: Integer. The official speed limit of that road.

As a convenience method we provided the method getDuration() which returned the time needed for a car to travel from the beginning to the end of the road with a speed equal to the speed limit.

The roads were also extracted from the Excel file and inserted into the DB by using the keys of the intersections.

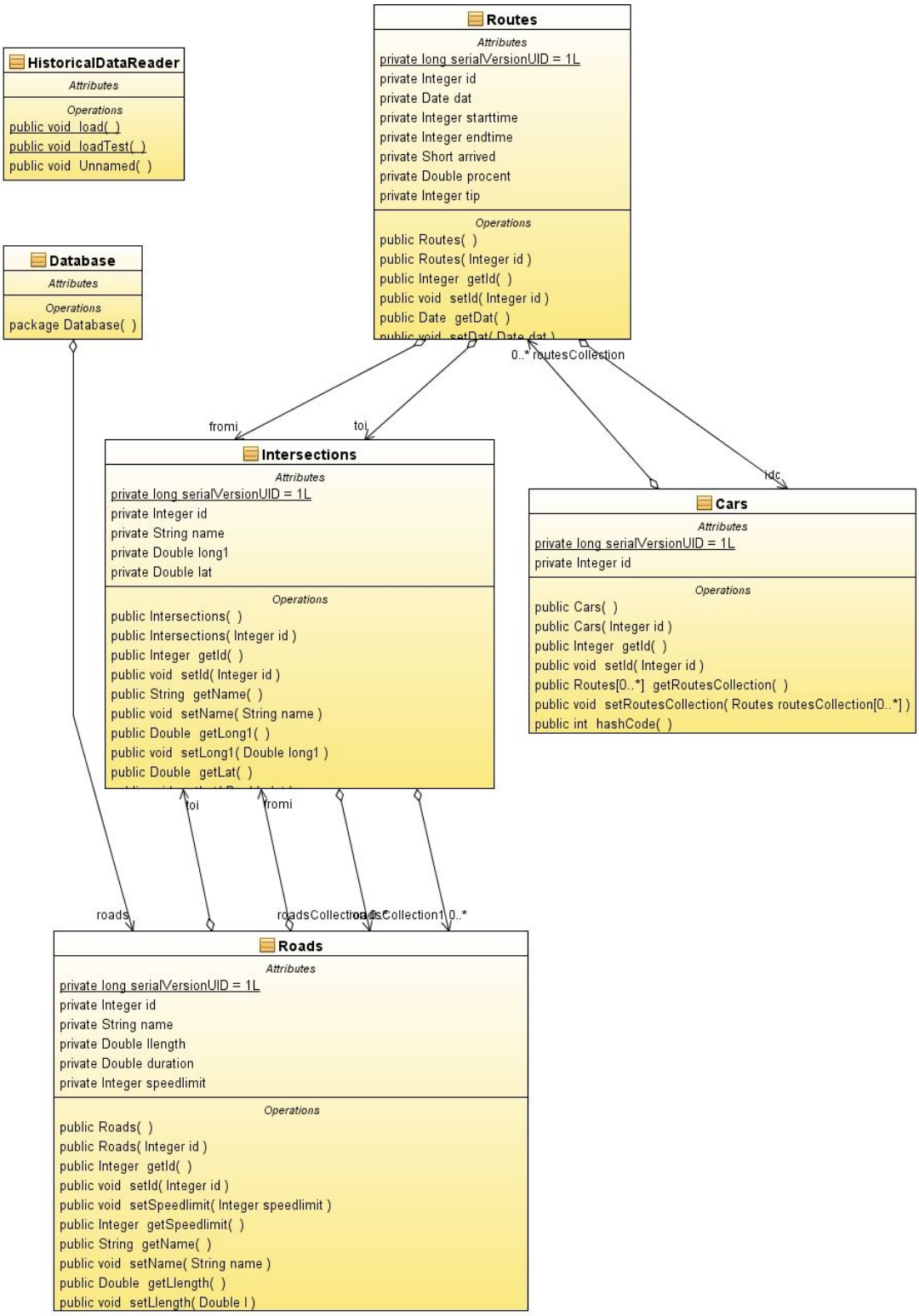


Figure 9.3: UML Class Diagram of the main data structures

Algorithm 4 Example of JPA annotation

```
@Entity
@Table(name = "INTERSECTIONS")
.....
public class Intersections implements Serializable {
  private static final long serialVersionUID = 1L;
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Basic(optional = false)
  @Column(name = "ID", nullable = false)
  private Integer id;
  @Column(name = "NAME", length = 255)
  private String name;
  @Column(name = "LONG", precision = 52)
  private Double long1;
.....
```

Time Intervals. The initial time intervals comprised in the historical data had a fixed sampling rate. More specifically, the day was split in intervals of 10 minutes. In our database we merged the intervals which had the same traveling time in longer intervals. This would eventually improve the search of an interval or the estimation of the traveling time in the application. The intervals for a road depend on the historical data or on the traffic model that can be applied to a particular day/situation.

The Intervals class has the following fields:

- IdRoad: Integer. The id of the road to which this time interval is associated.
- StartTime: Integer. Represents the time at which this interval starts in minutes.
- EndTime: Integer. Represents the time at which this interval ends, also in minutes.
- AverageSpeed: Double. Represents the average speed observed on the road at that time.

The UML class diagram presented in Figure 9.3 shows the database, the historical data retrieval and the most relevant data structures related to it. In the following paragraphs more details related to each main data structure are going to be presented.

9.4 Database Object Relational Mapping - using JPA

To ease the development we choose to use an object-relational mapping framework. Such a framework provides an easy way of defining the SQL tables, to add/remove data and to retrieve complex sets of data. As this is a very useful tool, it was standardized by JCP under the name of JSR 317. Java Community Process is a formalized process in which interested parties add standardized extensions to the Java language.

JPA provides a set of annotations to Java classes that are interpreted by external tools and provide the needed functionality. As an example for the class Intersections the annotations are presented in the piece of code at Algorithm 4. We can see many interesting aspects of the automatic mapping. We will explain some of these.

The 'Table' annotation specifies what is the name of the table which will store this class.

The 'Id' annotation, together with the 'GeneratedValue' and 'Basic' specify that the next field (id in our case) is an ID to the database (primary key in database language), which will be automatically generated and is compulsory.

Then 'normal' fields are mapped to columns using 'Column' annotation. For each an SQL type is specified.

The most complex part is the annotation that specifies the queries that can be executed for this class.

Besides this class another class named Controller is generated to help the loading of objects of this type. This class will use JPA's query language which is a simplified version of SQL to retrieve

Algorithm 5 Example of JPA query

```
Query q = em.createQuery("select object(o) from Intervals as o where o.idr=" + idr + " and o.starttime<=" +
time + " and o.endtime>" + time);
q.setHint("org.hibernate.cacheable", true);
return (Intervals) q.getSingleResult();
```

objects from the database. An example can be seen in the code at Algorithm 5. We can see a special notation to identify the objects to be returned - "object o". The rest of the command is valid SQL.

The framework implementing JPA standard used in this project is Hibernate. Taking advantage of the integration between Netbeans, JPA and Hibernate, most of the operations were done automatically or through a graphical interface (like configuring the connection settings from Hibernate to the database).

9.4.1 Database - Derby

The database that is associated to the application is a Java embedded DB. The Java DB is Sun's supported distribution of Apache Derby. Java DB is a fully transactional, secure, standards-based database server, written entirely in Java, and fully supports SQL, JDBC API, and Java EE technology. The Java DB database is packaged with the GlassFish application server.

The alternative to constructing a DB connected to the application was the data serialization. This approach has been used for the first prototype. However, there are certain disadvantages when using data serialization: it should ideally not be used with large-sized objects as it offers significant overhead and the Serializable interface does not offer fine-grained control over object access. Nevertheless, it is easy to use and can be customized.

Some of the advantages of having a DB are the following: reduced data redundancy, reduced updating errors and increased consistency, improved data security and improved data access to users through the use of host and query languages.

The DB contains the following tables: *intersections, roads, intervals, cars* and *routes.*

9.5 Client-Sever communication protocol

All the data that we need to display the roads will be kept in all the clients. This will allow the mobile clients to work even if the Internet connection is temporally lost. Still, the server has the most up to date traffic information. So if a route is needed this request has to be sent to the server and the response must be taken back.

As we need a simple communication we will use for this a simple TCP point to point connection. The client sends two type of messages:

- A status message reporting the position and the speed, so that the server can update the historical database. The message will be formatted as following: "pos,1,2,3.". "pos" is a fixed string which shows that a status message will follow. 1 and 2 represent the latitude and longitude (1 and 2 are given as examples). 3 represents the current speed in km/h (of course we would expect in a real situation to be a bit more than 3). The last point denotes the message has ended. The server doesn't give any response to this message.

- A query message, which asks for a route. The message is formatted as following: "req,1,2,600.". This message is composed of the source node id (1) the destination node id (2) and the departing hour in minutes from 0:00 (600 in our example). The server will then compute the route and respond with a message formatted as: "rou,1,4,5,6,2,700.". This encodes the route as a sequence of intersections and finishes with the time at which the algorithm estimates the traveler will arrive at destination.

We choose to use characters as this will avoid problems related to data type sizes and endianess on different devices. Also it helps a lot when developing the application to be able to read the messages.

9.6 Desktop Client User Interface

The GUI of the desktop client was implemented with the Swing GUI Builder (Matisse) in Netbeans. Initially, we developed a GUI that contained the basic functionalities of an interactive map. In order to do this we used the Open Street Maps API (discussed in Section 9.7) which allowed us to integrate a map and define various events. As we needed to select and draw our Intersections on the map rendered by OSM, we extended the component with the necessary functionality.

Therefore, the interface of the desktop client has multiple functionalities such as editing or requesting a route for both static and dynamic assignments. It also has options for inspecting the historical data.

The application starts with the main page which shows a large display of the map with an appropriate zoom level so that the whole country can be observed. On the map, the roads and the intersections contained in the database are graphically displayed (see Figure 9.4). The current date and time are also displayed in the left upper corner.

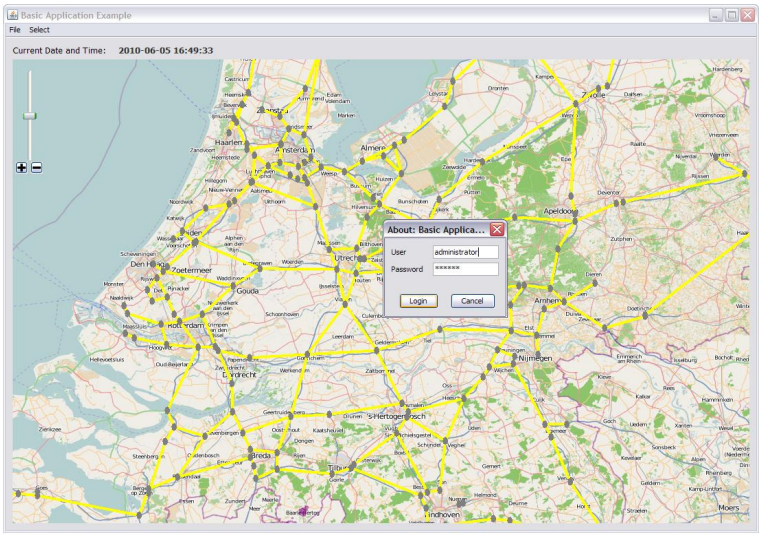


Figure 9.4: Interface - Main Page

The "File" Menu allows the user to choose between several options:

- *Login*: this option was created in order to be able to differentiate normal users from administrators of the application because of the editing option.
- *Edit Data*: this option leads the authenticated user to a new window which allows him to create new intersections and roads by linking the intersections. Moreover, a node or a road can be selected and/or deleted.
- *View Historical Data:* this option opens a new window which allows the user to get more insight into the historical data associated to each road. The purpose of this part is to offer an example on the variation of the average speed for a chosen road during a typical working day. Nevertheless, this part could be trivially extended to show an updated traffic model corresponding to the current day. In this way the roads on the map might be colored in a different color depending on the congestion level.

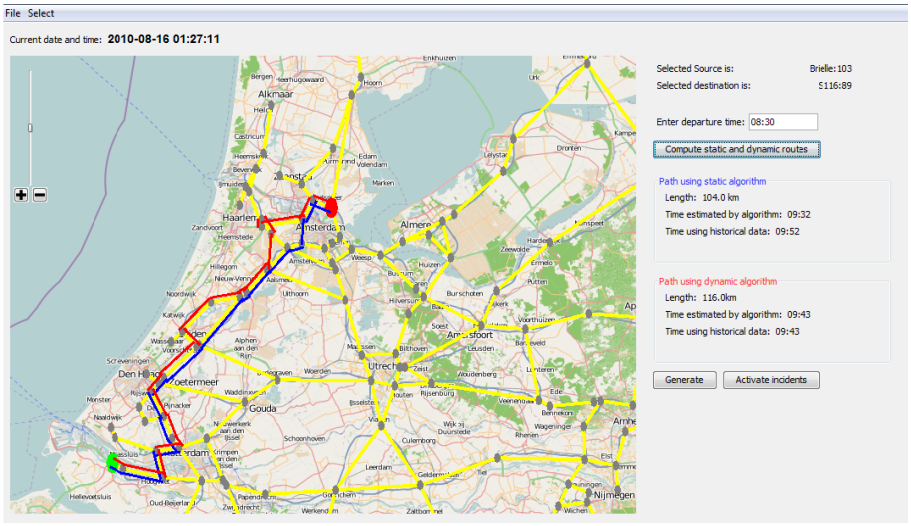


Figure 9.5: Interface - Routing Page

- *Start Routing:* this option allows the user to access the core part of the application, where he can ask for a route. We will describe the routing interface more in detail in Section 9.6.1.

9.6.1 Routing GUI

The routing GUI represents the most important functionality of the interface to the user. The user is requested to provide an origin, a destination and (optionally) a departure time for his trip. The default departure time considered by the algorithm would be the current time. Following this, a route is displayed on the map along with other details. In order to show which is the gain of the dynamic algorithm with regard to the travel time the static solution is also presented.

More in detail, the user can select a destination and a source node on the map by using the "Select" option from the menu. The selection of the node is done by clicking in the neighborhood of an Intersection on the map. This is implemented for example in the *DefaultMapController* object at the *mouse clicked* event. If x and y represent the map plane coordinates, $N = \{i \epsilon Intersections\}$, and $d[i]$ is the distance from the clicked point to the intersection. We also assume a threshold value for the distance between an intersection and a clicked node, δ. All intersections are retrieved from the DB in N and the intersection for which the distance to the clicked point is smaller than δ is considered selected. The procedure is presented in the piece of code in Algorithm 6.

The point selection on the map is more complex because there are also other cases in the interface where the click event is used so that we had to differentiate these cases. Such as the intersections selections in order to create a road is, or the intersection delete.

Algorithm 6 Intersection detection in mouse clicked event

For each $i \epsilon N$
$y' = OsmMercator.LatToY(i.getLat(), zoom)$;
$x' = OsmMercator.LongToX(i.getLong(), zoom)$;
$y' = map.center.y - map.getHeight()/2$;
$x' = map.center.x - map.getWidth()/2$;
$d[i] = \sqrt{(y'-y)^2 + (x'-x)^2}$;
if $(d[i] \leqslant \delta)$ then Intersection i is selected.

The names of the selected intersections are then displayed in the top right panel as *source* and *destination*. Both static and dynamic solutions are displayed in the right side panel along with the distance to be traveled, estimated travel time by the algorithms and the real traveling times, predicted based on the historical data.

Once the request for a road is transmitted the path is painted on the map. This is implemented in the *JmapViewer* object. Given the roads which are already painted on the map both the static and dynamic routes, in different colors are drawn at a precise distance ξ parallel to the road on both sides (see Figure 9.5).

Both pats (lists of intersections) are transmitted to the JmapViewer object. Here the plane map coordinates are computer for each two intersections along the route, belonging to a road by using the latitude and the longitude of the nodes. Given these coordinates, the plane coordinates of the new points for the parallel "roads" are computed as presented in the Equation 9.1 and Equation 9.2. Because a road is determined by a "from" and a "to" intersection these equations were used to determine the intersection from which the parallel road should start. These equations are used only for determining the bounding points for an edge contained in one of the parallel paths.

$$y'F = \xi/(1 + (yT - yF) * (yT - yF)/(xT - xF) * (xT - xF)) \tag{9.1}$$

$$x'F = -y'F * (yT - yF)/(xT - xF) \tag{9.2}$$

where $y'F$ and $x'F$ are the plane coordinates of the new point where from the edge starts; yT and xT are the plane coordinates of the ending intersection of the road belonging to the route; xF and yF are the plane coordinates of the starting intersection of the same road.

9.7 Open Street Map API

At the beginning several graph open source representation libraries (like JGraph etc) were studied because a solution that would provide the map was desired. But mostly none of these solutions provided a sufficiently permissive environment for building the Dutch highways network as similar as possible to the real one. For an *interactive user interface* the system required also a map such as other GPS systems have (TomTom, Garmin, Mio etc).

As OSM has a public available API that allows the necessary operations, this was studied further. The OSM API provides a number of utilities for manipulating maps (just like on the http://www.openstreetmap.org web page) and adding content to the map through a variety of services, allowing us to create robust map applications for our system. Thus, the OSM API allowed us embed an Open Street Map. In the remaining of the section we will briefly describe the implementation of the OSM map in our application.

JMapViewer is a Java component which allows to easily integrate an OSM map view into the desktop client. The UML class diagram of the JMapViewer is depicted in Figure 9.6. JMapViewer browses *map tiles* similar to the Slippy Map. Slippy Map is a term that refers to the main OSM map display, a web interface for browsing rendered OpenStreetMap data. It can cache tiles in the hard disk as well but our system renders the map tiles directly from the OSM server every time the application starts. So, JmapViewer came along with a lot of incorporated functionalities which were used in order to build up the user interface, for both the desktop and mobile client. For the mobile client we had to display the map differently but this is described in next section, Section 9.8.1. It also offered the possibility to write our own events and to add own graphical aspects such as the intersections, the roads or the routes on the map.

9.8 Mobile application

In this section we aim to present the way we rendered the map for the GUI of the mobile phone and to describe the use of the data structures at this level.

9.8.1 GUI

The mobile application is implemented using J2ME technologies. The main advantage is that as they use Java the transition was quite easy. Of course several components had to be simplified because of the limited resources available on such a device.

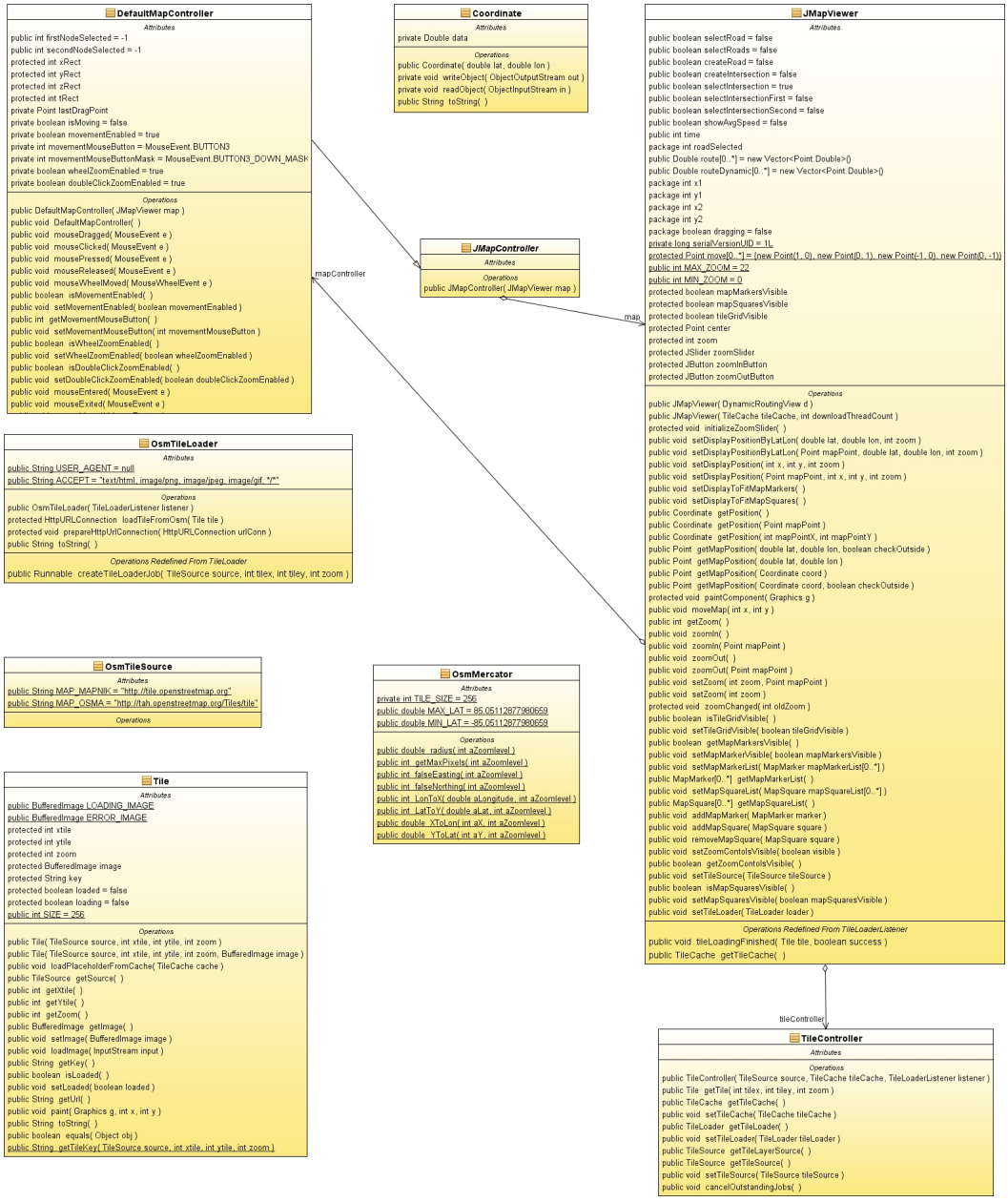


Figure 9.6: JMapViewer - UML class diagram

Algorithm 7 Connection to OSM server to render a map tile

```
public static int getXTile(double lon, int zoom)
{ return ((int) Math.floor((lon + 180) / 360 * (1 << zoom))); }
.....
public static int getYTile(double lat, int zoom)
{ return ((int) Math.floor((1 - log(Math.tan(Math.toRadians(lat)) + 1 / Math.cos(Math.toRadians(lat))) /
Math.PI) / 2 * (1 << zoom))); }
.....
HttpConnection c;
c=(HttpConnection) Connector.open("http://tile.openstreetmap.org/" + zoom + "/" + getXTile(lon, zoom) + "/"
+ getYTile(lat, zoom) + ".png");
c.setRequestProperty("Content-Language","es-ES");
c.setRequestProperty("User-Agent","Profile/MIDP-2.0Configuration/CLDC-1.0");
c.setRequestProperty("Connection", "close");
c.setRequestMethod(HttpConnection.GET);
int len = (int) c.getLength();
if (len > 0)
  {DataInputStream is = c.openDataInputStream();
  byte[] data = new byte[len];
  is.readFully(data);
  image = Image.createImage(data, 0, len);
  images.addElement(image);
```

One important component we had to do again was the Open Street Map renderer. At any point our application will draw at most 4 map tiles. Computing the position of these need to take into account the zoom level, dimension of the screen and current view position.

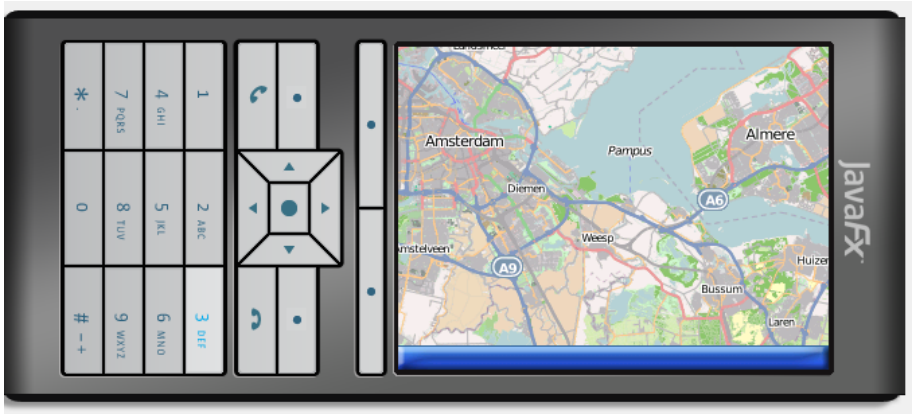


Figure 9.7: Mobile phone J2ME emulator - The process of rendering a map

The connection to the OSM server in order to get the map tile that we used for the left corner of the screen is presented in the piece of code at Algorithm 7. The numbel of a map tile is accessed by the String "http://tile.openstreetmap.org/" + zoom + "/" + getXTile + "/" + getYTile + ".png".

Loading new tiles is also a challenge as usually mobiles have a slower connection. To keep the application responsive we implement the loading in a different thread and display them just when they are fully loaded. However, the tiles which were already once loaded do not have to be loaded again from Internet.

9.8.2 Data

As the mobile device should be able to work even if the network connection goes temporarily down, we choose to have the network stored locally. Of course, given the device constraints i.e. a reduce

JVM we didn't use the full database that was used on the server. We just exported some of the data in text format, and reloaded it on the mobile.

Using this, after an initial connection to the server, the mobile can display the route. Even in extreme cases it can propose alternatives to the initial route, if there is no Internet connection to obtain a new, updated route from the server.

The full size of the network stored as text is 11kb.

Chapter 10

Conclusions and future work

The future work that we consider for this thesis mainly comprises the development of the incidents model. Ideas related to such an incidents management model are presented in Section 10.2.

Section 10.1 presents our conclusions about the main components that we developed in this thesis, by looking at the goals which we stated in the Introduction and the results which were obtained.

10.1 Conclusions

The research conducted in this thesis aimed to develop and implement a personal advanced traveling assistant software that guides the traveler from the start of his journey to the end offering him/her the most appropriate solutions. We started this ambitious project with many ideas that we intended to include and integrate in a successful application. This implied building a conceptual design and based on that we implemented a prototype. The first step was to perform a literature review that brought into light classical traffic models developed for dynamic traffic assignment. We examined state of the art personal traveling systems with a critical eye and tried to gather the best aspects and combine them into our project. We also tried to bring some new contribution for the transportation research field.

These were the generic steps that we followed in order to start the project but let's turn back to our goals stated in the introduction of this document and draw some conclusions.

10.1.1 Dynamic Traffic Assignment

As we mentioned in the introduction of this thesis, our main purpose was to build a dynamic traffic assignment system. An important component of this is the prediction of the traveling time. This prediction model is employed by a routing algorithm when computing the solution to a route request.

Prediction Model

In the thesis we proposed a prediction method that would update the historical data based on supervising the routes in the network together with the real time traffic information.

In order to develop the prediction model in our thesis we performed incremental improvements by building various models and adding to each model more and more advanced processings. By using this method, we could see the improvements of the results while the model was getting more advanced.

The prototype that we developed relies on real historical data. Because of the fact that sending out a great number of vehicles with a mobile phone that would report the position was not so feasible and also due to time constraints we chose to use just historical data for our prediction model. However, this gave a lot of insight into the dynamics of traffic and into the performance of the algorithm when using real variable traffic flows.

Dynamic algorithm

Our prototype implements an algorithm that is a time dimensional extended version of Dijkstra shortest path algorithm. The main difference is that our algorithm takes into account the traffic variations in time. The cost function in the algorithm is associated with the traveling time. As prediction model we used the one previously described, still any other prediction model could be easily used.

Because of the fact that the algorithm gives the route with the shortest time to each user we categorize it as a user equilibrium assignment. However, we assume that just a part of the drivers in the network are connected to the system. If we deal with the whole network the situation would change. A more detailed discussion on this subject is presented in Section 10.1.5.

The results of the algorithm were compared to the results of two versions of the static Dijkstra algorithm, one that computes the shortest path and one that uses the fastest roads given their maximum speed limit. Conclusions about the results of the algorithm are presented in Section 10.1.4.

10.1.2 Build a human-centered design

We developed a complex design for an advanced traveler information system that relies on the concept of a distributed systems. The system that we designed integrates the use of live traffic information that derives from tracking the individuals and use of the highway sensors. Travelers are routed through hand held devices which can be their mobile phones. An important feature of the system is that it is usable by everybody, without any special training or knowledge needed. In order to get more insight into users preferences with regard to such a system we did a user survey that mostly confirmed our expectations but also brought new ideas.

The system is also seen as a intelligent assistant as it has the capability to detect, learn the user's profile and associate it with his schedule. It combines this information with the traffic data and advises him about the best route to take.

Building the design of such a system was a challenging experience as there are numerous aspects to be taken into account. For each feature that we included in the design we also presented a possible manner to achieve it.

Given the huge complexity of the system we chose to implement the most important components with their basic characteristics.

10.1.3 Implement a prototype

We implemented an open system that includes the main functionalities assumed in the design. The purpose of the prototype was to offer a test bench for different algorithms, multimodal user interfaces, traffic data and eventually live traffic data.

We used open source tools for each component of the project and this has more advantages: we developed a portable application that can also be easily extended.

The prototype that we have built represents a working proof of concept for a dynamic routing assistant.

10.1.4 Tests of the algorithm

The algorithm was tested on real data collected from the highways of The Netherlands during 4 days. The results of the algorithm along with its strengths and weaknesses are presented in Chapter 8, Tables 8.1, 8.2, 8.3 and 8.4. We compared the results of our algorithm, DDTA (Dynamic Dijkstra Traffic Assignment) with the results of other two versions of the static Dijkstra algorithm, as we already mentioned.

To test the algorithms we chose to form routes between some major cities in The Netherlands. In order to create the routes we chose as origin and destination intersections on the ring of each city. The routes were requested at each 30 minutes during the whole day. Next, an average of the traveling time for all the routes connecting each pair of cities at every 30 minutes was computed.

The most interesting results were found, as expected, around the peak hours because otherwise the algorithm had either identical or similar solutions to the other algorithms. Following the computation of the traveling time we chose to show the average gains of DDTA in traveling time when compared to the shortest path static algorithm. But the most interesting part is the one that shows the difference that DDTA makes. This is represented by the maximum traveling time difference of DDTA when compared to the other algorithms. For each pair or cities, at every 30 minutes, we show which was the maximum difference in the traveling time. In this case for example we notice differences of even 25 minutes when traveling at 8 o'clock from Rotterdam to Amsterdam.

10.1.5 Discussion

In our experiments we proved by examples and results that dynamic routing makes a difference. Especially for peak congested hours we showed a significant reduction in the traveling time (Tables 8.4, Chapter 8). However, in the prediction model we did assumed that only some car drivers use our system, so that the flow of travelers and the corresponding travel speed along trajectories will be as defined in the historic data file. If many car drivers use our system this assumption will not hold. Then our dynamic traffic assignment system will impact the traffic streams, so that the travel speeds computed by historical data are not valid anymore. But for this we can use another experimental design, namely to track individuals. From the historic data file we are able to compute the traffic streams, that is to say for every traveler we are able to find out his starting and destination position. Moreover, when travelers use the mobile phone to get routes they input the starting and destination position. With this purpose we also integrated a position reporter in the mobile client that shall be used in order to get current updates on the traffic.

If we consider every traveler as an agent we get routed and non-routed travelers and we are able to compute the congestion by developing new models. We are able to attach storage capacity to every trajectory which will be dynamically filled. Next we can develop models computing the travel speed given the rest capacity. From the complexity of the proposed alternative it may be clear that this is beyond the scope of this thesis.

10.2 Future research

An incidents detection and management model is an important feature for the advanced highway traffic management systems.That is why we propose in this section a description of the model that we considered to integrate but due to time constraints and due to the complexity of each issue that we had to deal with, we let this for future work. The model is presented in Section 10.2.1.

10.2.1 Incidents Model

Nonrecurring congestion cannot be managed without real-time prediction. Incidents can create huge delays and form queues along the highways. They cause bottlenecks and sometimes even secondary incidents. In order to achieve a real-time incident congestion prediction it is necessary to both properly *detect* and *manage* incidents in real time.

The detection of incidents can be achieved by computing an average of the spped of the vehicles which send their position. If this average drops down unexpectedly, there is a problem. The incidents detection is a complex issue because there can be more factors to determine a sudden drop in the average speed. Moreover, incidents can be of different types: car collisions, a lorry block etc. We have to analyze the traffic parameters and assign the problem to a certain category. In order to classify the incident we propose a supervised learning method which analyzes the data and recognizes patterns, such as a neural network classifier or a SVM (Support Vector Machine) classifier. The classifier can be trained on associating traffic parameters to a certain type of incident.

The management of incidents assumes the estimation of the delay on the affected links and the computation of new routes for the vehicles heading in that direction. Thus, in order to detect the delay we need to compute the *size of the queue* that forms by taking into account the distance

between cars. The speed drops down directly proportional to the distance until the queue. It decreases abruptly close to the incident and then the impact is transmitted.

In the remaining of this section we will describe which are the incidents variables that the incidents model should take into account.

Incidents Variables

Incident Type. The type of the incident determines significantly the drop of the speed and the queue. It is essential to know whether all road is blocked or just some lanes. The evolution of traffic after an incident is dependent on how many lanes are affected. Moreover, it is important to know the duration of the incident. Roughly we can categorize incidents in the following cases:

- Cars collisions.The number of cars involved usually determine the number of lanes that are affected.
- An overturned lorry. This usually blocks the whole road.
- Temporary road maintenance fields. Although these are not incidents, if they are not planned from before they have similar consequences.
- Heavy weather conditions.

Vehicles density. The vehicles density change the outcome of incidents: there is a big difference if an incident happens during the day when there is a large volume of traffic or in the night when there are few cars.

Distance between cars. Distance between cars on highways is mainly dependent on the *speed* and *weather conditions*. During daylight with good, dry roads and normal traffic volume a recommended safe distance between cars is of 2 seconds (or the length of 4 cars). But this distance changes at different speeds. If cars enter a zone in which the average speed decreases the distance will also decrease because of the possibility of safely instantly breaking without hitting the next car. In heavy traffic, during night or when weather conditions are not ideal (eg. light rain, light fog or light snow) the recommended distance doubles at 4 seconds. Keeping a safe distance is important as most of the accidents on highways are determined by the vehicle in the back following too closely. If the speed is limited to 50 km/h then we may assume that there can be approximately 1 second between cars (or a two cars length).

This is an important variable that we would consider in our model in order to predict the decrease of the speed in time caused by the incident.

Bibliography

[1] M. Mirshahi, J. Obenberger, C. A. Fuhs, C. E. Howard, D. R. A. Krammes, D. B. T. Kuhn, R. M. Mayhew, M. A. Moore, K. Sahebjam, C. J. Stone, and J. L. Yung, "Active traffc management: The next step in congestion management," Tech. Rep. FHWA-PL-07-012, Offce of International Programs and Offce of Policy and Federal Highway Administration and U.S. Department of Transportation American Association of State Highway and Transportation Offcials, July 2007.

[2] F. Middelham, "Dynamic traffic management," tech. rep., Ministry of Transport and Public Works and Water Management, and AVV Transport Research Centre, Rotterdam, Netherlands, June 2006.

[3] A. Henk Taale, "Analysing loop data for quick evaluation of traffic management measures," 2006.

[4] B. S. Kerner, *The Physics of Traffic*. Springer, 2004.

[5] H. Rakha, M. Van Aerde, E. Case, and A. Ugge, "Evaluating the benefits and interactions of route guidance and traffic control strategies using simulation," *Vehicle Navigation and Information Systems Conference, 1989. Conference Record*, pp. 296 –303, sep 1989.

[6] H. Rakha and A. Tawfik, "Traffic networks: Dynamic traffic routing, assignment, and assessment," in *Encyclopedia of Complexity and Systems Science*, pp. 9429–9470, 2009.

[7] I. D. Greenwood and C. R. Bennett, "The effects of traffic congestion on fuel consumption," *Road and Transport Research*, vol. 5, no. 2, pp. 28–61, 1996.

[8] J. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings of the Institution of Civil Engineers, Part II*, vol. 1, no. 36, pp. 352–362, 1952.

[9] C. Fisk, "More paradoxes in the equilibrium assignment problem," *Transportation Research Part B: Methodological*, vol. 13, no. 4, pp. 305 – 309, 1979.

[10] M. J. Smith, "In a road network, increasing delay locally can reduce delay globally," *Transportation Research*, vol. 12, no. 6, pp. 419 – 422, 1978.

[11] N. F. Stewart, "Equilibrium vs system-optimal flow: Some examples," *Transportation Research Part A: General*, vol. 14, no. 2, pp. 81 – 84, 1980.

[12] G. B. Dantzig, "Discrete-variable extremum problems," *Operations Res.*, vol. 5, pp. 266–277, 1957.

[13] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, 1959.

[14] D. V. Vliet, "Road assignment 1 : Principles and parameters of model formulation," *Transportation Research*, vol. 10, no. 3, pp. 137 – 143, 1976.

[15] E. C. Matsoukis, "Road traffic assignment, a review, part 1: non-equilibrium methods," *Transportation Planning and Technology*, vol. 11, no. 1, pp. 69–79, 1986.

[16] M. V. Aerde and S. Yagar, "Dynamic integrated freeway/traffic signal networks: A routing-based modelling approach," *Transportation Research Part A: General*, vol. 22, no. 6, pp. 445 – 453, 1988.

[17] E. C. Matsoukis and P. C. Michalopoulos, "Road traffic assignment, a review, part 2: equilibrium methods," *Transportation Planning and Technology*, vol. 11, no. 2, pp. 117–135, 1986.

[18] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, pp. 95–110, 1956.

[19] L. J. LeBlanc, E. K. Morlok, and W. P. Pierskalla, "An efficient approach to solving the road network equilibrium traffic assignment problem," *Transportation Research*, vol. 9, no. 5, pp. 309 – 318, 1975.

[20] D. K. Merchant and G. L. Nemhauser, "A model and an algorithm for the dynamic traffic assignment problems," *Transportation Science*, vol. 12, no. 3, pp. 183–199, 1978.

[21] A. K. Ziliaskopoulos, "A linear programming model for the single destination system optimum dynamic traffic assignment problem," *Transportation Science*, vol. 34, no. 1, pp. 37–49, 2000.

[22] J. K. Ho, "A successive linear optimization approach to the dynamic traffic assignment problem," *TRANSPORTATION SCIENCE*, vol. 14, no. 4, pp. 295–305, 1980.

[23] J. R. Birge and J. K. Ho, "Optimal flows in stochastic dynamic networks with congestion," *Operations Research*, vol. 41, no. 1, pp. 203–216, 1993.

[24] M. Carey and E. Subrahmanian, "An approach to modelling time-varying flows on congested networks," *Transportation Research Part B: Methodological*, vol. 34, no. 3, pp. 157 – 183, 2000.

[25] T. L. Friesz, J. Luque, R. L. Tobin, and B.-W. Wie, "Dynamic network traffic assignment considered as a continuous time optimal control problem," *Operations Research*, vol. 37, no. 6, pp. 893–901, 1989.

[26] Hall, M. V. Vliet, and D. Willumsen, "Saturn - a simulation assignment model for the evaluation of traffic management schemes.," *Traffic Engineering and Control*, vol. 21, no. 4, pp. 168–176, 1980.

[27] N. B. Taylor, "The contram dynamic traffic assignment model," in *Networks and Spatial Economics, 3:297–322*, 2003.

[28] A. Abdelfatah and H. Mahmassani, "A simulation-based signal optimization algorithm within a dynamic traffic assignment framework," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pp. 428 –433, 2001.

[29] K. M. Lum, H. S. L. Fan, S. H. Lam, and P. Olszewski, "Speed-flow modeling of arterial roads in singapore," *Journal of Transportation Engineering*, vol. 124, no. 3, pp. 213–222, 1998.

[30] L. Kisgyörgy and L. R. Rilett, "Travel time prediction by advanced neural network," *Periodica Polytechnica Ser. Civ. Eng*, vol. 46, p. 2002, 2002.

[31] B. Coifman and M. Cassidy, "Vehicle reidentification and travel time measurement on congested freeways," *Transportation Research Part A: Policy and Practice*, vol. 36, no. 10, pp. 899 – 917, 2002.

[32] B. Coifman, "A new algorithm for vehicle reidentification and travel time measurement on freeways," pp. 167–174, 1998.

[33] B. Coifman and E. Ergueta, "Improved vehicle reidentification and travel time measurement on congested freeways," *Journal of Transportation Engineering*, vol. 129, no. 5, pp. 475–483, 2003.

[34] F. Dion and H. Rakha, "Estimating dynamic roadway travel times using automatic vehicle identification data for low sampling rates," *Transportation Research Part B: Methodological*, vol. 40, no. 9, pp. 745 – 766, 2006.

[35] A. Hobeika and C. K. Kim, "Traffic-flow-prediction systems based on upstream traffic," pp. 345 –350, aug-2 sep 1994.

[36] Y. Li and M. McDonald, "Link travel time estimation using single gps equipped probe vehicle," pp. 932 – 937, 2002.

[37] M. Burger, A. Hegyi, and B. De Schutter, "Suitability of different mean speeds for model-based traffic control," in *Proceedings of the 87th Annual Meeting of the Transportation Research Board*, (Washington, DC), Jan. 2008. Paper 08-2010.

[38] S. Ishak and H. Al-Deek, "Performance evaluation of short-term time-series traffic prediction model," *Journal of Transportation Engineering*, vol. 128, no. 6, pp. 490–498, 2002.

[39] J. Yu, G.-L. Chang, H. Ho, and Y. Liu, "Variation based online travel time prediction using clustered neural networks," pp. 85 –90, 12-15 2008.

[40] J. W. C. van Lint, "Reliable real-time framework for short-term freeway travel time prediction," *Journal of Transportation Engineering*, vol. 132, no. 12, pp. 921–932, 2006.

[41] J. van Lint, S. Hoogendoorn, and H. van Zuylen, "Accurate freeway travel time prediction with state-space neural networks under missing data," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 5-6, pp. 347 – 369, 2005.

[42] J.-B. Sheu, Y.-H. Chou, and L.-J. Shen, "A stochastic estimation approach to real-time prediction of incident effects on freeway traffic congestion," *Transportation Research Part B: Methodological*, vol. 35, no. 6, pp. 575 – 592, 2001.

[43] A. Garib, A. E. Radwan, and H. Al-Deek, "Estimating magnitude and duration of incident delays," *Journal of Transportation Engineering*, vol. 123, no. 6, pp. 459–466, 1997.

[44] N. B. Hounsell and S. Ishtiaq, "Journey time forecasting for dynamic route guidance systems in incident conditions," *International Journal of Forecasting*, vol. 13, no. 1, pp. 33 – 42, 1997.

[45] L. Rothkrantz, D. Datcu, and M. Beelen, "Personal intelligent travel assistant, a distributed approach," in *ICAI 2005*, 2005 June.

[46] M. Broucke and P. Varaiya, "The automated highway system: A transportation technology for the 21st century," *Control Engineering Practice*, vol. 5, no. 11, pp. 1583 – 1590, 1997.

[47] L. Baskar, B. De Schutter, and H. Hellendoorn, "Hierarchical traffic control and management with intelligent vehicles," in *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium (IV'07)*, (Istanbul, Turkey), pp. 834–839, June 2007.

[48] K. Li and P. Ioannou, "Modeling of traffic flow of automated vehicles," vol. 5, pp. 99–113, June 2004.

[49] S. Tsugawa, S. Kato, T. Matsui, H. Naganawa, and H. Fujii, "An architecture for cooperative driving of automated vehicles," *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pp. 422 –427, 2000.

[50] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz, "Cartalk 2000: safe and comfortable driving based upon inter-vehicle-communication," *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, pp. 545 – 550 vol.2, june 2002.

[51] R. C. Sondak, "Dynamic intelligent algorithm for navigation," Master's thesis, Delft University of Technology, Man-Machine Interaction Group.

[52] M. Maguire, "Methods to support human-centred design," *International Journal Human-Computer Studies*, vol. 55, pp. 587–634, 2001.