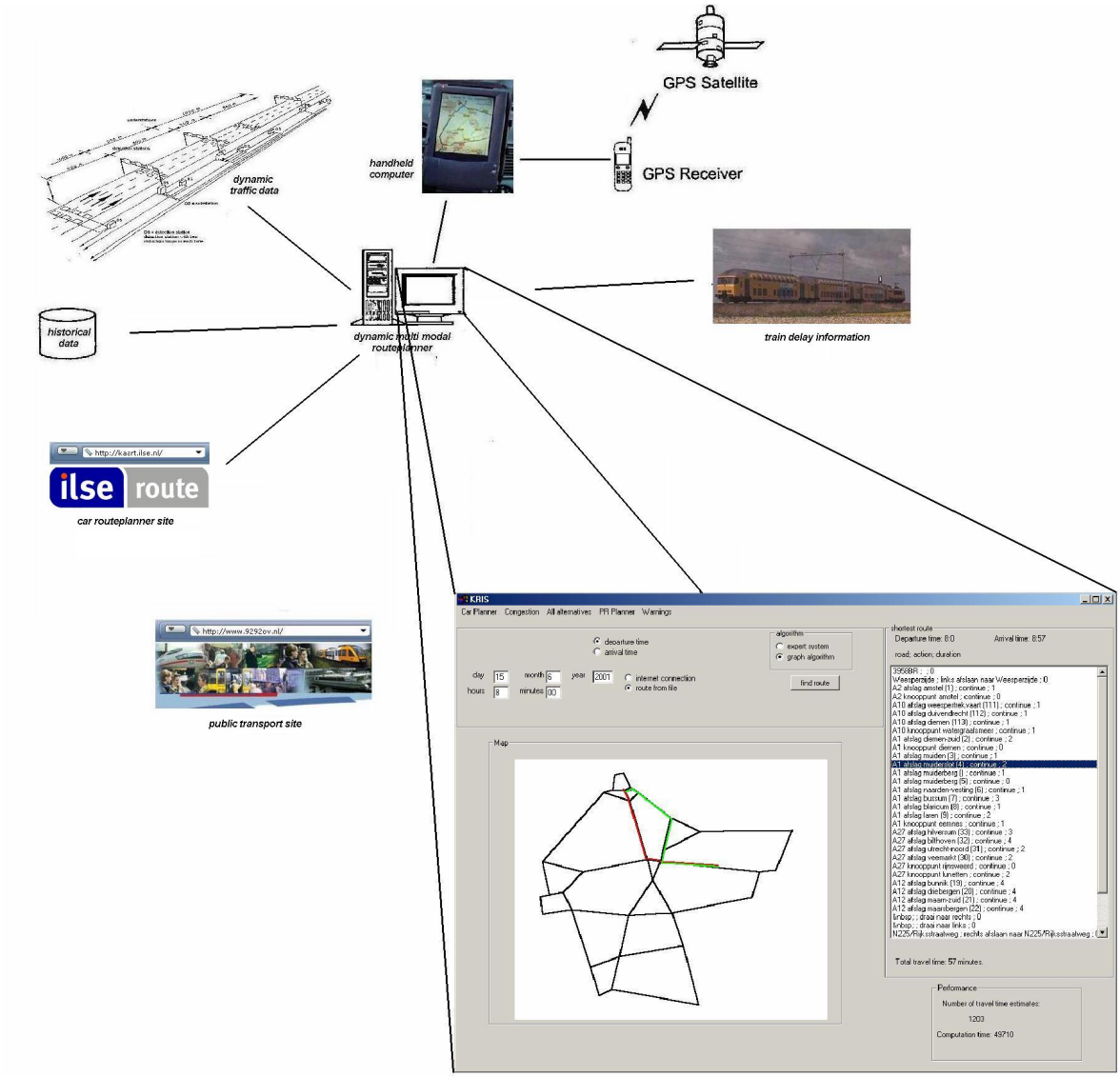
DYNAMIC MULTI MODAL ROUTE PLANNING:

AN ARTIFICIAL INTELLIGENCE APPROACH



Master's thesis of Gerritjan Eggenkamp

Delft University of Technology
Faculty of Information Technology and Systems
July 2001

TU Delft

Graduation Committee:

Dr. drs. L.J.M Rothkrantz
Prof. dr. ir. E.J.H. Kerckhoffs
Prof. dr. H. Koppelaar (chairman)

Eggenkamp, Gerritjan (G.Eggenkamp@twi.tudelft.nl)

Master's thesis, July 2001
Dynamic Multi Modal Route Planning: an intelligent approach.

Delft University of Technology, The Netherlands
Faculty of Information Technology and Systems
Knowledge Based Systems Group

Keywords: Artificial intelligence, Dynamic route planning, Multi modal route planning, Intelligent agents, Expert systems, Personal route guidance.

Preface

This master's thesis describes the research I have done to graduate at the Knowledge Based Systems (KBS) of the Delft University of Technology, headed by Prof. dr. H. Koppelaar. This research was done within the Netherlands Research School for Transport, Infrastructure and Logistics, TRAIL, headed by Prof. dr. Ir. P.H. Bovy. Within TRAIL research is carried out into Seamless Multi Modal Transportation (SMM): the utopia in transportation, where travellers are guided during their journey and never have to wait when changing transportation mode, since a central planner knows their destination and allocates transportation means to handle all transportation demands. As we all know this utopia is still day-dreaming, since at the moment it seems even impossible to carry out the train schedules according to plan.
During this graduation research it was tried to get one step further towards the SMM utopia: a multi modal route planner was designed that takes into account all known delays. Although no allocation according to the current travel demands is done yet, the individual traveller is advised the best route possible given the current situation, without having to worrying about what route or transportation modality to choose.

Project

Most people who know me quite well will acknowledge the fact I am always preoccupied with finding the best way to perform a number of different tasks on a day, the shortest route to get somewhere, the most optimal configuration of green-times of traffic lights and other optimisation problems like these. When my supervisor Leon Rothkrantz told me about the research that was carried out within the SMM program, I was caught immediately.
During the first months I found out many possibilities can be explored in the area of guiding travellers and it was hard to focus towards one particular subject. With the help of Leon Rothkrantz I decided to direct the research at the development of a dynamic multi modal route planner, since no such route planner was yet available. Especially the fact several information sources had to be combined in an intelligent way made such a planner a challenging project: humans are somehow capable of finding the best route and transportation modes to their destination using different information sources (train tables, heuristics about which trains often are delayed, car planners, traffic jam reports, etc.) so the challenge of this project was to develop a system that performs the same intelligent behaviour. The results of this research are summarised in this report.

Acknowledgements

First of all, I would like to thank Leon Rothkrantz for his enthusiastic guidance and advice during this project. I really enjoyed our (two) weekly sessions and I am glad I listened to his advice to focus towards one project in stead of trying to explore everything I found interesting. Especially our 'invention' of the extended-Dijkstra algorithm, which proved to be only a little bit different than the Dijkstra algorithm itself when we reviewed it, is a good example of these inspiring meetings.
Secondly, I would like to thank Niels van der Zwan and Peter Kluit, who helped me developing the object-oriented model. It was interesting to work two days at IBM in Uithoorn and the help Niels provided me during my first steps in the object-oriented world were very useful. Peter Kluit helped me very much in finding the right (OO-) architecture for the reasoning modules and his sharp questions and remarks made me able to pinpoint and solve crucial problems in this architecture.

Thirdly, I would like to thank Boi Sletterink, who helped me several times when I encountered Java problems. But especially his help in ‘hacking’ the public transport site was essential, since otherwise no multi modal route planner would have been running now. Furthermore I would like to thank Robert van Vark, for his help and guidance on the PITA.
Last, but definitely not least, I would like to thank my family and especially Christine Vink, for their moral support and understanding for my continuous occupation with KRIS.

Gerritjan Eggenkamp,
Amsterdam, July 2001.

Summary

Currently no dynamic route planners are present that take different transport modes into account. Although the highway network in the Netherlands is flooded with cars and is subject to heavy congestion in both rush hours and about 22% of the trains is delayed no planner is available that finds the fastest route using all kinds of transportation and considering all occuring delays.
In this thesis state of the art applications in this field are reviewed and the research that has been carried out into such a route planner is described, culminating in the design of such a planner and the implementation of a prototype, called KRIS (Knowledge based Routing Information System). It is shown that such a planner is feasible, although its quality will depend on the availability of public transport delay data and sophisticated travel time predictors for the roads in the highway network.
When research was carried out to the performance of traditional shortest path algorithms, like Dijkstra's algorithm, in such a route planner, it showed that the computation time increases significantly when dynamic data are incorporated. Consequently, other possibilities were explored and since humans are quite well capable of finding alternative routes in the case of congestion it was decided to study the feasibility of an expert system approach. In the prototype that has been build two implementations have been made, one using an expert system approach, the other using a Dijkstra-like shortest path algorithm, to be able to compare both methods.
It is concluded that the expert system approach shows great potential. Not only requires the expert system significantly less computation time, it is also possible to perform incremental searching: alternative routes are computed one by one and the best one found so far is available at every moment. The incremental searching property of the algorithm may prove to be very useful in a real time application, for example when a user is approaching a junction and needs to know quickly which route to take.

Table of Contents

Chapter 1: Introduction

This study explores the possibilities of a dynamic multi modal route planner. In the first section an introduction to the problem setting will be given. Then the context in which the research was done is presented (section 2 and 3), introducing TRAIL and the Personal Intelligent Travel Assistant. In section 4 a definition of a dynamic multi modal planner is given and its properties are discussed, while in section 5 the goals of this project and the methods used are being discussed. The last section, section 6, gives an overview of the structure of this report.

1.1 Problem setting

Nowadays, especially in the Netherlands, the road network is flooded by cars. In the rush hours a total of 200 kilometres of traffic jam is reported every day. The direct costs of the delays caused by these queues have been estimated to be about 1.7 billion guilders in 1998, whereas the indirect costs may be even higher. These costs include follow-up costs because of arriving early or late at the destination, and prevention costs of trying to be in time [bovy00]. In Figure 1.1 some key congestion figures for the Dutch motorways are summarised.
Consequently, a lot of effort is carried out to solve these traffic problems. Research into automatic car following systems is carried out, which aims to reduce the minimum distance between cars and consequently to increase the capacity of the road network. Teleworking at home is encouraged, to reduce the number of movements each day. Large sums of money are invested in the public transport system and the road network. Simultaneously monitoring systems are placed to track the traffic on the motorway network.

15	locations of permanent peak hour queue building
50	queues each working day
> 1	hour average queue duration
200	kilometers of daily queue length
20%	of peak hour drivers in Randstad area end up in queues
100,000	hours travel time loss daily

Figure 1.1: Congestion in the Netherlands [bovy00].

Another approach to reduce travel time for individual travellers would be to develop a route planner that incorporates traffic information and public transport possibilities when searching the fastest route: if congestion occurs along the normal route, this planner will search for the best alternative, which may lead the traveller along other highways, but may also advise the traveller to drive to the nearest train station to continue his trip using public transport. Thus the individual traveller's time is reduced significantly, but also the overall load on the road network is spread, which results in less congestion.
Unfortunately, such route planners do not yet exist. Currently available route planners have two serious drawbacks. Firstly, most of these route planners take no dynamic traffic information into account. Considering the very high traffic density on the Dutch road network this is a serious demerit. These so called 'static planners' only provide optimal routes when there is no congestion along the planned route. As was stated before, every working day there is an average of at least 200 kilometres of congested road, so these route planners are only useful during relatively few hours of the day. It would be of great help in planning a trip if the planner could incorporate dynamic traffic information. Not only would such a planner be able to search for another route when there is congestion along the statically planned route, it would also give a realistically estimated travel time. Thus a

traveller would know when to leave from home and would be able to compare between different kinds of transportation.
The second major drawback of currently available route planners is, that there is no planner available, in which the use of different kinds of transportation is possible. There are many route planners for cars available and there is a route planner for the public transport system, but there is no planner available which investigates the possibility of combining car and public transport. When congestion occurs along a vital road of the route and no reasonable road alternatives are available for example it may be a good alternative route to pass this congested road by train. Thus one could first travel by car, while there is no congestion, and then change modality to train when congestion occurs along the desired route.
Considering the above mentioned points it may be clear that a planner capable of searching alternative routes using traffic data on the one hand and of integrating different (transport) modalities on the other hand, would be very useful.

1.2 Seamless Multi modal Mobility

Since this research was conducted within the research program “Seamless Multi modal Mobility” (SMM) of the Netherlands Research School for TRAnsport, Infrastructure and Logistics (TRAIL) an introduction to this research school and the research program will be given in this section.
TRAIL is the Dutch academic research institution targeting world-wide developments in transport, infrastructure and logistics and combining top-level education and research. A knowledge institute in which the Delft University of Technology (DUT), the Erasmus University of Rotterdam (EUR) and the University of Groningen actively participate. Faculties of economics, business administration, mathematics and technology are among those combining their strengths to create unprecedented levels of knowledge synergy. The purpose: using experience and talents combined with those of clients to offer solutions to challenges in these fields across the world TRAIL Research School is a breeding ground for innovative solutions [trail99].
The research program “Seamless Multi modal Mobility” (SMM) of the Netherlands Research School for Transport, Infrastructure and Logistics (TRAIL) at the Delft University of Technology aims at developing new public transport services for the next century. The new transport services will be developed based on the concept of chain mobility. In this concept, a chain conductor is responsible for the perfect connection between all different transport modalities. The traveller will be transported from door to door without spending time on finding the travel schedule best suited to his needs or worrying about possible delays or calamities while being in transit. Applying the results of the SMM program to current public transport systems will result in drastic changes in these transport services as new transport modalities and transfer nodes will be introduced and transport services will be provided on a more demand-driven and less fixed basis [trail99].

1.3 Personal Intelligent Travel Assistant

Within the research program SMM research is carried out into the Personal Intelligent Travel Assistant (PITA). The research into the development of a dynamic multi modal route planner was done within the context of the PITA. In this section an introduction will be given into the PITA. An extensive overview of the architecture of the PITA will be given in chapter 3.
The PITA is a handheld device that provides ubiquitous communication between travellers and service providers, like the Dutch railways and bus companies. This communication consists of travel planning and reserving capacity before travelling.

The PITA will integrate several information services, ranging from the complex task of planning a travel schedule for a future trip using different transport modalities to informing the user of transport delays and alternative travel schedules.

While travelling, the PITA will guide the traveller by signalling upcoming transfer points and providing information on transfer routes in the nodes. En route, the PITA will also provide the traveller with information concerning delays, calamities and alternative routes, if applicable.
The sentence in italics denotes the functionality of the PITA that inspired this research. The potential of a dynamic multi modal route planner is only utilised to its fullest extent when used within a PITA (the terms dynamic and multi modal will be clarified in section 1.4). When used in combination with a PITA, the PITA can monitor the progress of the chosen route and detect delays. As soon as delays are noticed the planner starts to search for alternative routes, which are communicated to the traveller by the PITA.
When a multi modal route planner is used autonomously the traveller will consult the planner before leaving. Consequently the traveller will travel along the route that was best at the moment of departure. As a result of the dynamic behaviour of the underlying system this route may not be the best anymore when the traveller is already on his way. For example, an accident might happen along the route, which would result in a traffic jam. As was already mentioned in the first section of this chapter a dynamic multi modal route planner would be able to construct a new route that might pass the congested road by train or by using another carroute. The traveller can be informed about this delay and the alternative route by the PITA, while otherwise he would have had to monitor the situation on the road and rail network himself, for example by calling the traffic information service and the public transport companies, and would have to construct alternative routes himself.
It should be noticed that the aims of the SMM program and the PITA are different, although both projects are strongly dependent of each other. The SMM program is aimed at optimising travel objectives for all travellers or a large group of travellers, while PITA's goal is to optimise the journey of an individual traveller. For example, the SMM program will develop services that compute the optimal assignment of public transport vehicles based on the travel demands of all travellers, while the PITA will provide the individual traveller with the optimal route for him. Since optimal assignments can only be made when knowing each individual travellers wishes the PITA is necessary in the SMM program, since it can provide the needed information.

1.4 Dynamic multi modal route planning

In the previous sections sometimes the words 'modality' and 'multi modal' were used. Also the word 'dynamic' was used. Both terms will often be used throughout this report, since they actually denote the essence of the route planner that was constructed: a dynamic multi modal route planner. Considering this the meaning of these two words will be clarified in this section. After this clarification also an introduction into dynamic multi modal route planning will be given.
The word 'modality' is used in the fields of philosophy, psychology and music, but is not incorporated in the Oxford dictionary or other general dictionaries. In specific philosophy and psychology dictionaries it can be found that the word modality is based on the Latin word 'modus', which can be translated into 'measure', 'manner', 'means' or simply 'mode'. A multi modal journey thus is a journey that uses different modes of transport, while a multi modal route planner is a route planner that finds the optimal route using different kinds of transportation.
In the context of the PITA and the route planner that is being developed, it should be noticed the term multi modal is also used in another context, the interface. When humans are interacting, the sender can use different muscle production signs (modalities), like speech,

articulation, typing, etc., to show his message, while the receiver can use his five senses (modalities), sight, hear, touch, taste and smell, to receive the message. Since the PITA is not a human being artificial modalities should be used to send and receive messages, like speech recognition, short messages, etc.. In section 3.3 the main modalities the PITA uses are identified.
The meaning of dynamic is denoted as follows in the Oxford dictionary:

dy-na-mic: adj. **1** (*approving*) (of a person) forceful, and having a lot of energy: *a dynamic personality* **2** (of a process) always changing and making progress **OPP** STATIC **3** (*physics*) (of a force or power) producing movement **OPP** STATIC.

The second meaning is most applicable in the context of route planning: it concerns the dynamics of the process of transportation. As opposed to static route planning, dynamic route planning is concerned with route planning that takes into account the 'always changing' property of transportation.
Combining the previous meanings of the words dynamic and modality a 'dynamic multi modal route planner' can best be expressed as:

> *A route planner that takes into account different kinds of transportation while incorporating historic and real-time traffic data of these different kinds of transportation.*

Combining different transportation modalities and historic and real-time traffic data results in an explosion of the search space. Firstly, on the contrary to static route planning now the departure time influences the travel time. Consequently, for each trajectory that is taken into account, travelled by train, car or any other modality, the travel time has to be known for every minute of the day. Secondly, different kinds of modality are taken into account. Of all available transportation modalities the time tables (in the case of public transport) and real time traffic data have to be taken into account. One can imagine the enormous explosion of the search space due to these two aspects, compared to the search space of currently available route planners. Commonly used algorithms to find the shortest route cannot be used anymore as a result of the explosion of the search space, since the performance of these algorithms will degrade significantly.
Somehow all available information, like time tables, distances between junctions, historical traffic data, real time traffic data, information on delays of the trains, etc., has to be combined and has to be reasoned with. It seems that human beings are quite well capable of deciding which modalities to use for their trip and which route to choose, using static route planners for car and public transport, heuristic knowledge about congestion and delays of trains, traffic jam reports, etc., especially when they are familiar with the trajectory (for example the daily route to their work). Apparently it is possible for humans to reason using these facts. This suggests artificial intelligence can be used to construct a multi modal route planner that integrates dynamic traffic data.
In Figure 1.2 a schematic overview of the complex task such a dynamic multi modal route planner should perform is given. All different kinds of available information should somehow be combined to find the optimal route for a traveller. A handheld device coupled to a Global Positioning System (GPS) receiver could be used to find the location of the traveller and the handheld device itself could be used to make a route request. Using traffic information, historical data, information about train delays, static public transport and car planners the planner should find an optimal route. These data are acquired by consulting webpages from car route, public transport route and queue report providers using search agents.

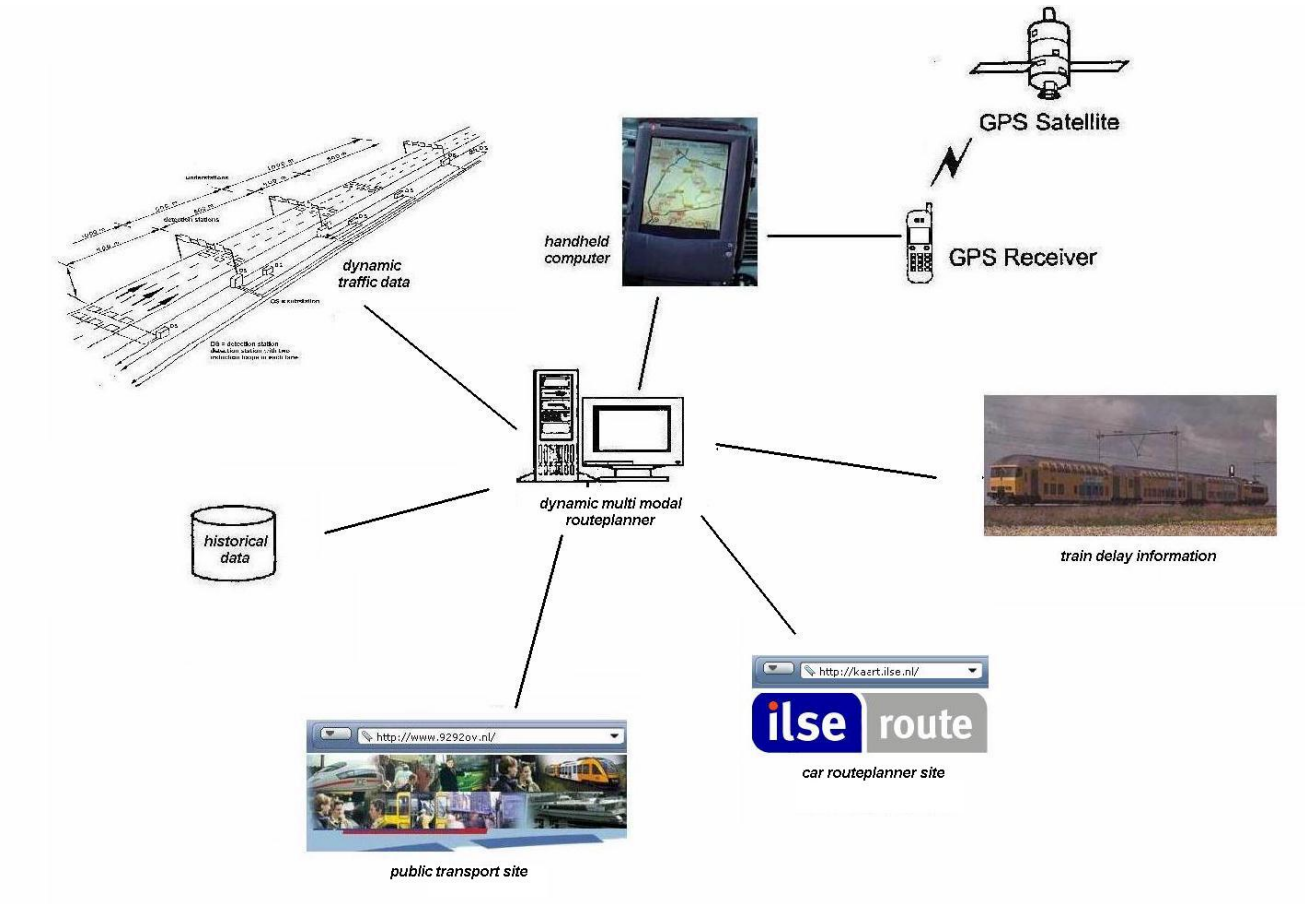


Figure 1.2: Possible information sources a dynamic multi modal route planner could use to find the optimal trip.

1.5 Goals

In the previous sections the need for a dynamic multi modal route planner was identified. This resulted in the first goal of this research: to study the possibilities of a dynamic multi modal route planner. If possible a design should be made and a prototype build.
In section 1.4 it was also suggested artificial intelligence might be used in a dynamic multi modal route planner, since the performance of 'traditional brute force' algorithms might degrade significantly. The second goal of this research was inspired by this suggestion: to study the feasibility of an artificial intelligence approach and to compare this approach to dynamic route planning with the traditional brute force algorithms. To be able to compare these approaches both approaches should be incorporated in the prototype.
It was chosen to research the possibilities of an expert system, since there are many experts available and an expert system can imitate the reasoning process of experienced travellers. Travellers have their own expertise on how to travel their daily routes, which can be used by the expert system. Other AI techniques could also be used, but showed many implementation problems. Consequently, in this research the feasibility of one specific AI technique (expert system) that seemed the most promising in the field of route planning is researched. In further research the feasibility of other techniques could be studied.
There are also many related fields where artificial intelligence can be used. For example in the field of travel time estimation, where neural networks seem very promising [lint00]. During this project we focused on one main aspect: route planning.

Summarising the goals of this thesis, these can be stated as follows:

- To make a design of a multi modal route planner,
- to develop a prototype that is able to find the best (multi modal) route by using dynamic traffic data, using an expert system approach,
- to develop a prototype that is able to find the best (multi modal) route by using dynamic traffic data, using an algorithmic approach,
- to make a comparison between these two approaches, especially with respect to the performance and quality of found routes.

An acronym was chosen to be able to identify the dynamic multi modal route planner in a more convenient way. Because of the intelligent approach that is applied in this study and the task of combining different information sources this acronym became “Knowledge based Routing Information System”, KRIS.

1.6 Structure

The report is divided into three parts. The first part focuses on the context within which KRIS was developed, namely traveller guidance systems. The second part covers the design of KRIS, the multi modal route planner that was constructed. The last part gives an introduction into the working of KRIS and discusses the test results. It also contains the conclusion and recommendations. In the last three paragraphs of this section the chapters of each part will be shortly introduced, but first a remark about the structure of the chapters is made.
If applicable each chapter is divided into two parts. Each chapter starts with a theoretical part, followed by a description of the application of this theory onto KRIS. Of course, this division is only made if there is theory appropriate to the subject of the chapter. It was chosen to discuss applicable theory at the beginning of each chapter in stead of in a different part or chapter of this thesis, because of the kaleidoscopic character of the theory that had to be discussed. In this way the context is explicitly shown, while otherwise a chapter of isolated theory issues would have been constructed.
Chapter two, which is the first chapter of part one, will introduce ‘state of the art’ concepts and applications concerning dynamic travel information and routing. An overview of currently available software and hardware is given. Chapter three gives an overview of the structure of the Personal Intelligent Travel Assistant. The research described in part two of this report was initiated as the route planner for this PITA. Before the development of KRIS started the architecture of such a PITA was explicitly stated and the results of this phase can be found in chapter three. Actually, the PITA can be seen as a motivation for KRIS.
The second part starts with an overview of the design of KRIS. Chapter four covers the requirements, while chapter five introduces the algorithm that is developed to construct multi modal routes. Two implementations of the algorithm were constructed, an expert system and a graph approach, which are discussed in chapters six and seven. Finally, in chapter eight, the problem of estimating travel times is discussed.
The last part of this report discusses the results KRIS produces. First, in chapter 9, the program itself is introduced and evaluated considering the requirements that are given in chapter 4. In chapter 10 a comparison between the expert system and graph approach is made. This thesis ends with a conclusion and recommendations.

PART I:

ROUTE GUIDANCE SYSTEMS

Chapter 2: State of the art applications

In this chapter currently available applications with respect to route planning are identified and examined. Firstly an introduction is given in section 2.1. Following this introduction available route planners and car navigation systems are examined (sections 2.2 and 2.3). Then more elaborate systems are identified. In section 2.4 the Travelstar is introduced, a Dutch initiative to build a dynamic car route planner, while in section 2.5 a similar German product is introduced, the Tegaron Scout. Finally, in section 2.6 the PROMISE project is described; an European project that aims to develop a personal travel assistant.

2.1 Introduction

Different applications are available at the moment in the field of dynamic multi modal route planning. Unfortunately, as was mentioned in the previous chapter, there is no application available that integrates both aspects: dynamic data *and* different modalities. Multi modal planners are available, but none using dynamic data. Car navigation systems exist, but no one offers true dynamic planning and (as could be expected) they do not take other transportation modalities into account. In this chapter an overview is given of state of the art applications that are currently available.
First available route planners and their properties are identified. These planners include car, public transport and multi modal route planners. Secondly, an overview is given of available car navigation systems, since one would expect these navigation systems incorporate dynamic data in some way. Following the general section on car navigation systems the Travelstar car guidance system is introduced. This product was the result of an initiative of the Dutch Ministry of Public Works and was developed especially to provide dynamic route planning. Another car navigation system that is discussed is the German dynamic route planner of Tegaron, as far as known the only planner that offers true dynamic routing. The last section introduces the PROMISE project, an initiative of the European Commission to provide travellers with a direct and easy access to multi modal travel and traffic information during their whole journey: actually a system with the same requirements as the PITA.

2.2 Route planners

At the moment several static route planners are available both on CD-ROM and the Internet. More then 80 car planners are available, the national railway company offers a train planner and a route planner for all public transport modalities is available. There are also two multi modal route planners available.
The car planners all require a departure and destination address and compute a fastest route. Other ‘best’ route objectives often cannot be chosen, although sometimes the cheapest or shortest route (in kilometres) can be planned. No available route planner incorporates dynamic traffic information. This means that if the shortest route is heavily congested these systems still route travellers along this congested route, because it is the shortest one. In [pop00] an extensive overview of available route planners is given.
The planner for the public transport is being exploited by a consortium of public transport companies, called OVR. It offers travel advice from door to door, although no other public transport modality then walking is taken into account to get to a public transport stop, for example using a bike or car. This planner incorporates dynamic data slightly: the railroad works of the following 24 hours are taken into account.
At the moment there are two route planners that are actually multi modal, the Planner Plus and the RouteKompas. Especially the RouteKompas can be named a true multi modal route planner, since it incorporates all different kinds of transportation. The Planner Plus only uses

the modalities train, car and walking. No other public transport modalities are taken into account. Unfortunately, both planners do not take dynamic data into account.

2.3 Car navigators

In the Netherlands, different car planners are available. To name a few, VDO offers the VDO Dayton navigator, Philips offered CARiN (now sold to VDO Dayton), Souren offers the CarNavigator, while Alpine, Magellan and other companies have navigators named after their company name. Most car navigation systems are build-in systems, using detectors on the wheels to calculate the distance driven and the turns made, and consequently are very expensive. At the moment cheaper navigation systems become available that are not build-in and use a Global Positioning System (GPS) device in combination with a laptop or palmtop computer with electronic maps. An example is the CarNavigator from Souren that uses a laptop attached to the dashboard in combination with a GPS receiver. The GPS unit is connected to the laptop using an interface that translates the input to be used by the navigation software.
As far as known there is no route planner available that offers true dynamic planning. Some route planners offer some kind of dynamic planning, by letting the user select roads that can be blocked. Using the traffic information that is available on the radio or by phone they can block the roads that are congested and get alternative routes. Of course this process can be automated and probably this will be available soon. It should be noticed different alternatives could not be compared this way, since the roads that are blocked are simply not taken into account. o comparison can be made between a route using the congested roads and other roads and it may well be possible the traveller is advised a route that takes longer then it would have taken if he would have travelled along the road that was congested.

2.4 Travelstar

In the introduction of this thesis it was already mentioned a dynamic route planner would utilise the available capacity of the road network better, which would result in less congestion. The Dutch government also recognises this fact and initiated the so-called ‘Realisatie In-Car’ (RIC) project. The goal of this project is to develop an in-car navigation system that guides the traveller based on static and dynamic information. The Traffic Message Channel using the Radio Data System (RDS-TMC) provides the dynamic data. This radio channel exists since two years and dispatches dynamic traffic information [reijm99, sto00].

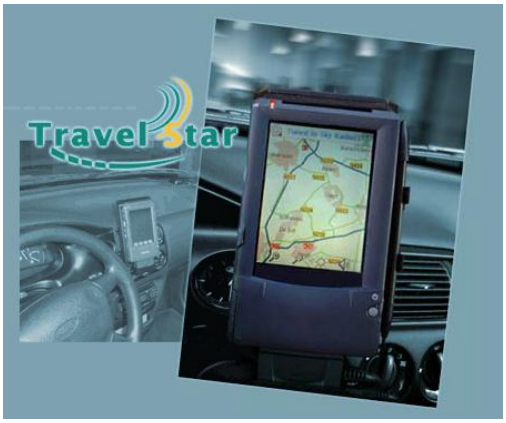


Figure 2.1: The Travelstar [trav].

The Ministry of Public Works (Ministerie van Verkeer en Waterstaat) put out to contract this project to the company Ars T&TT. Ars T&TT developed the Travelstar, a hand-held device that can be placed in the car. In Figure 2.1 the Travelstar is shown.

The hand-held device is a HP Jornada palmtop computer with touch screen, which can also be used outside the car (as a palmtop). The HP Jornada is connected to a GPS receiver, which is also placed in the car. Ars T&TT developed software that translates the RDS-TMC messages to visual objects on the screen. On the basis of the GPS location the screen 'moves' along the route the car drives, showing congested roads ahead. At the moment no dynamic route planning is offered: the Travelstar is only capable of planning statically and showing a map displaying the congested roads. Ars T&TT mentions it wishes to develop a dynamic route planner, but it is not clear when this planner will become available. During the year 2000 the Travelstar was placed in 1000 cars to get user feedback. In the summer of 2001 the Travelstar should become available for sale [trav].
The strongest aspect of the Travelstar is its price. Apart from the HP Jornada, which actually is a palmtop computer and costs about a 1000 guilders, the Travelstar only costs about 900 guilders. Compared to other navigation systems this is a very low price. Another advantage of the Travelstar is its compatibility. The Travelstar can easily be taken out of a car and placed in another car and the Travelstar does not require an additional communication system to receive dynamic information: instead the RDS-TMC channel is used.
The main disadvantage at the moment is that no dynamic route planner is available yet. A second drawback is the fact the RDS-TMC messages about congestion have a significant delay. The RDS-TMC messages give the situation of the road network of at least 10 minutes ago and only provide queue length in kilometres, while travel times would be more useful, since it is not possible to deduce travel times from queue lengths [sto00].

2.5 Tegaron Scout

The Tegaron Scout is a product of the German company Tegaron Telematics GmbH. At the moment the Tegaron Scout is the only in-car navigator known that actually plans using dynamic data. Like the Travelstar the Tegaron Scout also uses a palmtop computer, the Compaq iPaq, but the communication is done in a completely different way. The iPaq is connected to a mobile GSM telephone and a GPS device. When starting a journey the traveller gives his destination. The Tegaron Scout sends this route request to the central Tegaron station, using the mobile telephone, where the best route is computed using available dynamic data. The computed route is send back to the iPaq that stores the route and gives the driver the directions to follow using the GPS position. This is illustrated in Figure 2.2. Tegaron asks 2 D-Marks for each route request.

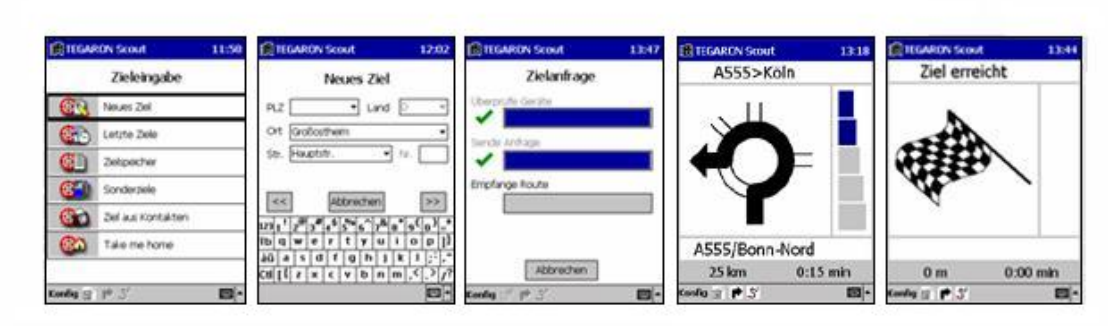


Figure 2.2: Different screen views of the Tegaron Scout [teg].

It should be noticed the route is not updated to new congestion when the traveller is on the way. The only way this can be done is by making new route requests when on the way. The initiative to do this is completely in the hands of the user, since the central station does not keep track of travellers that are on the way and whose routes might be disturbed. This aspect can be seen as the major drawback of the Tegaron Scout. To overcome this problem the Scout has an option of automatically checking for new congestion along the route, but each update costs another D-Mark.

Main advantages of the Scout are, of course, the fact dynamic data is taken into account and the fact only a small investment in equipment has to be made (as with the Travelstar). The advantage of the architecture of the Tegaron Scout (central computation) is that also other equipment can be used to get a route advice, for example using a WAP or using speech recognition. At the moment also route advice using a WAP telephone is available from Tegaron.

2.6 PROMISE

The Personal Mobile Traveller and Traffic Information Service (PROMISE) project was initiated by the European Commission. The PROMISE project partners included some leading companies of telecommunications, car and electronics manufacturing and map and information technology in Europe: Nokia, Volvo, IBM, Renault, BMW and others.

Country	Site	Special Characteristics
Finland	Helsinki and E18 Finland	Road weather information, public transport trip planners
Sweden	Gothenburg	Real-time information of urban transport
The United Kingdom	Scotland	Road information, SCOTIA
The Netherlands	Eurodelta	Interurban Traffic information, development of Dutch TIC
France	Paris and Ile de France	Urban traffic and public transport information,
Germany	Bavaria (Munich)	Links to German projects

Figure 2.3: PROMISE test sites [oj99].

The aim of the PROMISE project was to provide travellers with a direct and easy access to multi modal traveller and traffic information during their whole journey through mobile phones and hand-held PCs with wireless data communication. The project ended in February 1999.
The PROMISE partners build up test systems with various services. Technical functionality and market readiness was assessed through extensive verification and demonstration with hundreds of European test services. A demonstration phase was been carried out in six countries. In Figure 2.3 the different test sites and their characteristics can be found [oj99].

Figure 2.4: The Nokia 9000i Communictor.

At all test sites the Nokia 9000i Communicator was used as a hand-held device. Basically all the services were world-wide-web based, with a special (WAP-like) interface to the Nokia Communicator. Services like a route planner, an overview of traffic reports, public transport schedules, etc., all could be accessed through the Communicator. In some test sites the actual position of the user was used to show a map of the neighbourhood, possibly showing the nearest public transport stops.

Although stated as the major aim of the project, no dynamic multi modal route planner was (yet) developed. The traveller still has to combine traffic reports with planned routes himself. The PROMISE project made these services available at a hand-held device, but currently available WAP telephones make this Internet services also available.

Chapter 3: Personal Intelligent Travel Assistant

As was stated in the first chapter, this research was carried out within the context of the Personal Intelligent Travel Assistant (PITA). Although the route planner that is constructed also might be used apart from the PITA, in this chapter the PITA is introduced. First an introduction is given into the tasks the PITA should perform and the relationship between the PITA and the multi modal route planner. Then an overview of the architecture of the PITA is given. Section 3.2 introduces the design of the PITA. The sections that follow each summarize the different modules of this design. In section 3.3 the dialogue manager that takes care of the communication is discussed. The agent that monitors the journey and that handles the user profiles is introduced in section 3.4. The planner itself is reviewed in section 3.5, while the change module that detects delays is discussed in section 3.6. In section 3.7 the issue of a centralized or decentralized architecture is addressed and the last section explores the possibilities of following the positions of the users.

3.1 Introduction

In the first chapter already a short description of the PITA was given. Summarised in one sentence, the PITA should guide a traveller during his journey from the departure to the destination address. During this journey the traveller should not worry whether he travels along the best route: the PITA guides the traveller completely, also when unforeseen events (accidents) happen. It should be noticed that the best route is the best route with respect to the travellers wishes. Most travellers want to travel along the fastest route, but other travellers may want to have as little transfers as possible, to travel along the cheapest route or may have another interpretation of best route.
As was already stated in section 1.3 a dynamic multi modal route planner is only used to its fullest extent when used in combination with the PITA. Consequently a significant part of the research carried out in this project has been spent in studying the design of the PITA. The research carried out within the SMM programme so far was mainly focused on the dialogue management between the traveller and the PITA [vark00]. Only little research was done into the overall design of the PITA and especially into the routing part [yil00].
In this chapter a design of the PITA is introduced which could be used in combination with the dynamic multi modal route planner. A summary is given of each module in this design. The summaries give an overview of the functionality of the module and the degree of development.

3.2 Design

In Figure 3.1 the architecture that was designed for the PITA can be found. In the following sections the main modules of this design, the dialogue manager, agent, planner and change module are discussed. In this section the design is illustrated by the actions the PITA takes to guide a traveller during his journey.
When a traveller wants to make a journey, he contacts the agent by using the dialogue-manager. Then the agent makes a route request to the route module, which forwards the request to the planner. Using travel date and time, departure and destination address, available modalities and a best route objective, the planner starts to plan. The planner will send different possible routes (using different modalities) to the route module. Now a route has to be chosen. An intelligent agent can perform these tasks, with or without communication with the traveller. Using a user profile which is based on previous choices,

actual wishes and requests of the traveller, the agent chooses the best option with regard to the preferences of the traveller.
Now a route is chosen, the actions that the agent has to initiate to inform the traveller during his journey are put into the action-list. The number and kind of actions the agents has to initiate depend on the user profile. For example, an experienced traveller only wants to know whether he has delays, while a novice traveller that almost never uses public transport wants to be informed about every detail of the journey: in which stations the train stops, where he has to change trains, which way he has to walk towards the buses, etc. A timer keeps track of the time and sends the actions out of the list to the agent, when the agents should initiate any actions.

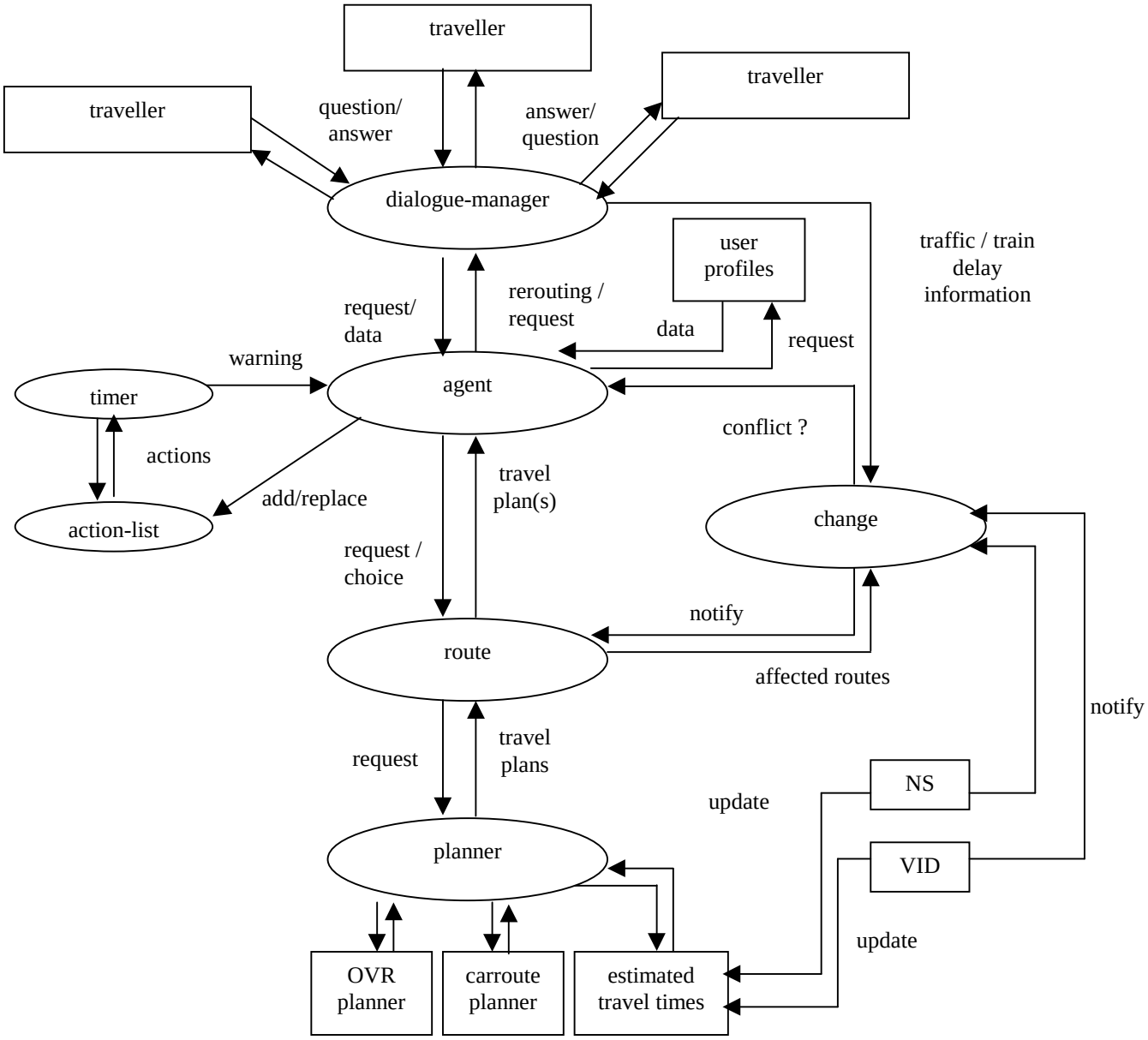


Figure 3.1: the PITA architecture.

Unfortunately, journeys are often subject to delays or congestion. When information about delays or congestion becomes available, the change module is being informed. A message to the route module, where all ‘current’ journey-routes have been stored, is being sent,

consisting of detailed information about the delay or traffic jam. The route module examines if there are routes that will be influenced by the delays and if so, it sends a new request to the planner to see if there is a better alternative route. The change module also sends the information to the travel time estimation modules, so when the planner is planning a new route, the new information is taken into account.
When an alternative route is found the new route is sent to the change module, which forwards the new route to the agent. The agent replaces the items in the action-list by the items of the new route and the dialogue manager informs the traveler.

3.3 Dialogue manager

The dialogue manager takes care of all communication between the PITA and the traveller. The focus of the PITA project was (until now!) on developing a multi modal dialogue manager. As was stated in chapter 1, in the context of the dialogue manager multi modal refers to different dialogue media. For the moment the following dialogue modalities are foreseen in the PITA [vark00]:

- ***Short Message Service (SMS)***: short message service is an information service that can be found on many platforms for mobile telephony. It can be used to sent text messages with a maximum size of 160 characters to other mobile subscribers or information servers.
- ***Wireless Application Protocol (WAP)***: the wireless application protocol combines the best of two rapidly evolving network technologies: mobile data and the Internet. Most of the technology developed for the Internet has been designed for desktop and larger computer, medium to high bandwidth and reliable data networks. WAP is a protocol set making Internet-like information services available for handheld devices.
- ***Automated spoken language***: automated spoken language interfaces are used to interface information systems by using the human voice. Spoken language interface can be used in virtually any environment, although adverse environments, like driving a car at high speed, cause a significant drop in performance. Over the last decade, speech recognition has evolved dramatically making it applicable to a wide range of applications, among which (complex) information retrieval tasks.

A coding scheme for managing PITA dialogues has been developed. The coding scheme is being validated at the moment, after which new management strategies for information dialogues will be developed. These new strategies must lead to flexible and adaptive dialogue control in order to facilitate the wide range in task complexity, user experience and interface functionality in the domain of Personal Intelligent Travel Assistants. An extensive description of the research into the dialogue manager can be found in [vark00].

3.4 Agent

As was stated in section 3.2 the agent can be seen as the spider in its web: the agent has to initiate all actions. Two methods can be chosen to implement the agent. Firstly, an (intelligent) agent can be assigned to each traveller, which keeps track of the progress of the journey of the traveller. When the traveller arrives at the destination the agent is deleted. In [kro99] this approach was implemented. This approach resulted in a poor performance of the system, since at every moment all agents have to keep track of delays, progress, etc.. Consequently another approach was designed and the timer, action list and the change module were introduced, which warn the agent if changes are observed in the originally planned route. This can be done in a centralised or de-centralised way: each traveller can have its own agent that also has his own timer, action list and change module and there can

be one agent that monitors the progress of all travellers that use the PITA. In section 3.7 this issue is addressed.
The agent also keeps track of the history of the different users (travellers). In Figure 3.1 this is illustrated by the storage of user profiles. These user profiles can be used to personalise the routes the PITA proposes to the travellers. For example in the user profile it might be stored if a traveller is able to run for a train or walks very slowly; information that can be used to determine whether travel schedules are feasible. A one minute transfer from two platforms that are 250 meters apart can only be made by a traveller, who is able to run. Using user profiles it becomes also possible to learn the preferences of travellers; whether they prefer to travel by car or public transport, whether they prefer to travel a longer distance without congested roads or shorter distance in traffic jam, whether they prefer a longer train trip without transfers or a shorter one with a few transfers, etc.. As is illustrated by the previous examples, a lot of possibilities of personal route guidance become feasible using user profiles. In [vark00] an extensive overview of these possibilities is given.
It should be noticed the agents make the travel requests that are send to the dynamic multi modal route planner that is studied in this project. Consequently, the agent has to infer from the user profiles, which parameters that have to be optimised (the objectives) are given to the route planner. According to these parameters the planner starts to search for the cheapest route, shortest route, route with the least transfers, etc..

3.5 Planner

The planner receives route requests from the agent. As was stated in the previous section these travel requests are personalised by the agent. The agent infers the objectives that have to be optimised and the planner returns the best route given these objectives. The multi modal route planner that is being studied in this project *could* be used as the planner used by the PITA, since it aims to give a personalised route advice. The planner could also be used stand-alone *and* the PITA could use another planner. Since the following chapters of this thesis are all dedicated to this planner in this section no further introduction will be given. This section will end with a remark on the route planning of travellers that travel the same route each day.
The largest part of travellers is commuters. They travel the same route every day. In the currently proposed approach every day a route request would be done for these travellers and the best route would have to be found by the planner. This could also be done in a different way. The route planner could generate all possible alternative routes, which are stored in the user profile of the traveller. Every time the traveller has to travel between these two addresses all routes are being updated for their objectives (travel time, number of transfers, costs, etc.) and the best route is chosen according to the objective the traveller prefers. This way no searching has to be done every time: the only thing to do is to update the values of the different objectives for the different trajectories of each route.
In Table 3.1 such data are given. In the right part of the table (columns 6 to 9) the possible routes are given. The ones in the rows of these columns denote if a link belongs to a route. The different links are given in the fifth column. Examples of links are the train connection between Delft and Rotterdam, the highway between the ramp Delft-Zuid and Rotterdam-Centrum, the route from the Julianalaan 1 to the ramp Delft-Zuid, the bustrip from the Julianalaan 1 to Delft station, etc.. The right part of the table is only changed if routes are changed or if new routes become available (then a new column is added). The left part of the table is constantly updated according to the latest information. In Figure 3.2 the travel time, costs, percentage of the link that is along highways and the chance at a delayed train are the objectives on which each link is being judged. Of course a lot of other objectives can be used too. Every time the travel time changes, the first column is updated and every time the costs change the second column is updated, etc..

A comparison of the routes for a given objective can now be made by adding the values in the objective column for each row where the route has a one. For example, the travel time for route 1 is 28 + 18 + 8 = 54 minutes, since link A, C and E belong to route A. Subsequently the costs of route 4 are fl. 12,-- + fl. 3,25 = fl. 15,25.

Table 3.1: An example of a route objective table.

Travel Time	Costs	Percentage highway	Chance at delayed train	Links	Route 1	Route 2	Route 3	Route 4
28 min.	Fl. 12,--	18%	8%	Link A	1	1	1	1
24 min.	Fl. 8,50	100%	0%	Link B		1		
18 min.	Fl. 5,75	0%	34%	Link C	1	1	1	
9 min.	Fl. 3,25	0%	12%	Link D				1
8 min.	Fl. 5,25	12%	5%	Link E	1	1	1	

3.6 Change module

Since the performance would degrade significantly if the agent was constantly monitoring the progress of the journey (see section 3.4) changes on the routes had to be detected by another mechanism. To detect these changes the change-module was designed. As long as no significant changes occur in the underlying network the change module remains silent, but immediately after a change, for example due to an accident, a delayed train, etc., the change module notifies the agent about the delay. A separate module should assess changes if their impact is large enough to start the notifying process: of course small changes, like a delay of a train of less then a minute, should not invoke a complete change procedure. Now the agent investigates which routes are affected by the change and will give route requests to the route planner. This process is called re-routing. Since the traveller is already on his way these routes have a new starting point.
This ‘re-routing’ can be done by sending a new route request to the planner with the current position as the departure address. Since the change module also updates the dynamic travel times this re-routing will be done according to the latest information.

3.7 (De) centralised approach

In the design that is shown in Figure 3.1 a central agent is monitoring all travelers. The traveler communicates through his cellular phone or other device with this agent. Another approach would be to let each traveler have his own agent on his own handheld computer. Since handheld computers become cheaper each day it may economically be more efficient to use this architecture. Still communication is needed in this approach, since the changes in the network have to be send to all agents. In Figures 3.2 and 3.3 both architectures are illustrated. It should be noticed that a decentralized approach can be compared to most car guidance systems which have their own maps and computational power and receive traffic jam reports by radio (see also section 2.3).
The centralized approach has as its major drawback that the whole system will stop working if the central computer has a problem. In a decentralized approach, the handheld PITA’s will keep functioning when the central computer goes down. Especially, if different bases have been installed which provide dynamic information: if one of them stops functioning, the others still keep working and the handheld PITA’s keep functioning correctly. On the other hand the decentralized approach requires a lot of message passing: every change has to be send to all handheld PITA’s, since no central computer is available that keeps track of Of course some kind of hybrid structure can be chosen, for example where each central computers keeps track of some of the PITA (for example the PITA’s that are nearest to him) and decides which information is send to them. If one of these centra does not function any

more, the other start sending all information. To summarize the points mentioned in this section, in Table 3.2 the qualities of both approaches have been listed.

Table 3.2: Properties of centralized and decentralized approach.

Centralized	Decentralized
No change messages Low reliability Route message sending No local computation power needed Easy updating of time tables and road and public transport network Very powerful mainframe needed that performs all route planning	Much change message sending Good reliability No route sending; local computation Local processor needed The local computers have to be updated in case of changed time tables or networks Only change messages have to be made centrally: only small central computation capacity needed

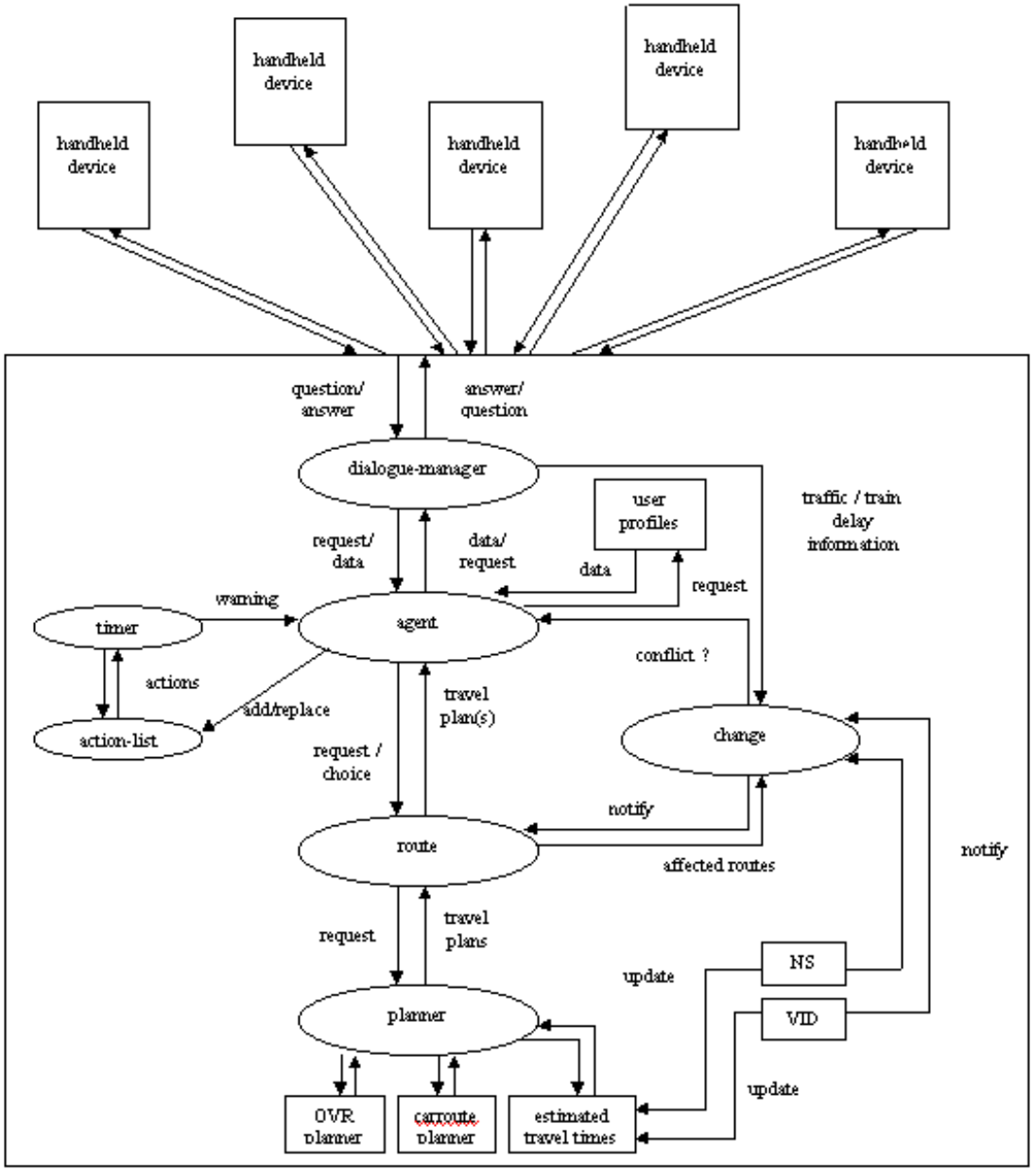


Figure 3.2: Centralized architecture.

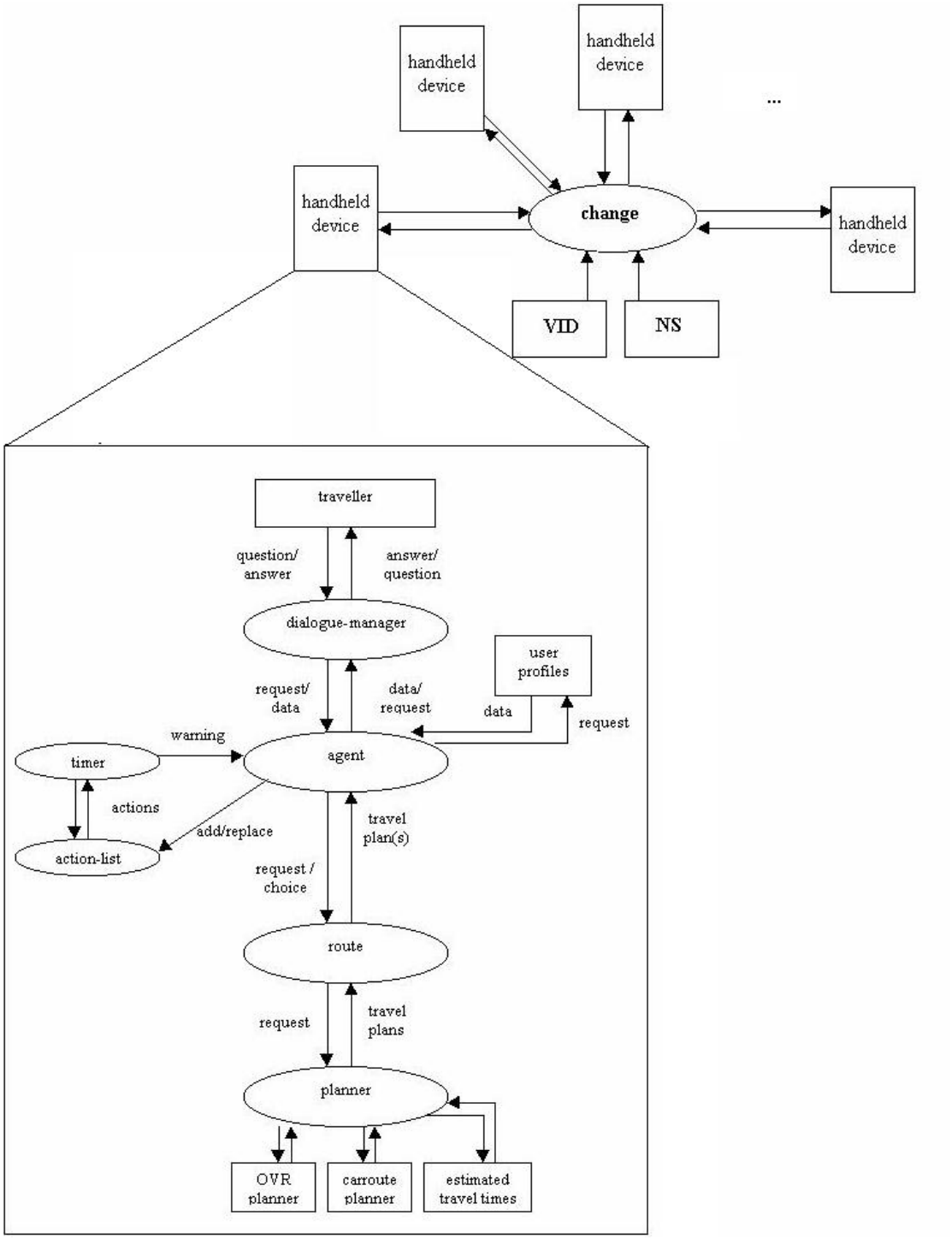


Figure 3.3: Decentralized architecture.

3.8 Using user positions

So far nothing was remarked about the way dynamic data should be collected. In Figure 3.1 the national Railway Company (NS) and the national information service on traffic jams (VID) have been identified as information providers to the change module and the route planner. It may be clear that the success of the PITA depends largely on the quality of the dynamic data that are used. When out of date information is used travellers may miss trains since their own train was delayed, may get stuck in a traffic jam, etc.. Consequently, good contracts should be made with the information providers.

Another approach would be to collect the dynamic data using the travellers that are connected to the PITA. When the position of each traveller is monitored and compared to the position they should be according to their travel plan changes can be identified. Using these differences the state of the underlying network can be updated. Several methods can be used to monitor the position of the users:

- Global Positioning System (GPS) : each traveller could be equipped with a hand-held device that also contains a GPS module. Using this module the exact location of the traveller can be identified and this location can be sent to the PITA central computer. If the PITA is distributed and is running on the device the traveller possesses only changes are being reported.
- Global System for Mobile Communication (GSM) : it is also possible to determine the position of a traveller by using the GSM network. Measuring the intensity of the signals of the different ground stations the positions relative to these stations can be found.
- User feedback : the user can be asked to give a message (press a button, dial a number, etc.) when he arrives at a station, junction or any given point. Now the time of arrival can be compared to the expected arrival time.

A major benefit of this approach is that the PITA becomes independent of the information providers. On the other hand enough travellers are needed in this approach to guarantee the working of the PITA: when there are too little travellers no information is obtained about the status of the network and nothing can be said about the travel times of alternative routes. Of course, some kind of hybrid architecture can be deployed where the PITA functions autonomously when enough travellers are connected and switches to information obtained from other sources when not enough data are available.
Actually four levels of extended functionality can be noticed. In Table 3.3 they are illustrated. At the zero level a PITA using static data can be found. The PITA just advises routes according to the time tables and ‘free flow’ travel times along the road network. No information about traffic jams, delayed trains, closed roads, etc. is incorporated. One level higher a semi-dynamic version of this route planner can be found: the same planner is used, but traffic information is incorporated by simply blocking the roads that are congested. This way an alternative route – avoiding all congested roads – is found, although this route may not be faster. At the second level the dynamic version can be found. On this level the PITA incorporates data about the current status of the underlying network in its travel advice. At the highest level a completely autonomous functioning PITA can be found, which infers dynamic data from the positions of its own users.

Table 3.3: Levels of functionality.

Level 0	**Static route planning**	No dynamic information is taken into account.
Level 1	**Semi-dynamic route planning**	Roads that are subjected to congestion are simply seen as blocked roads. Alternative routes that do not use these roads are found, although they may be slower.
Level 2	**Dynamic route planning**	Dynamic information is taken into account.
Level 3	**Interactive dynamic route planning**	Dynamic information is derived from the user positions.

PART II:

DESIGN

Chapter 4: Requirements

This chapter will identify the requirements of a dynamic multi modal route planner. In the first section the functionality of KRIS is divided into the functionality of three modules. In the following three sections (sections 4.2 to 4.4) the requirements for each of these modules are stated. In the next chapter the algorithm that was designed to meet these requirements is discussed.

4.1 Different modules

The requirements of KRIS can be split into the requirements of three main parts of the system. In Figure 4.1 these different main modules are illustrated. Of course, the main module is the route planner itself, which will be described in section 4.2. The requirements that have to be stated about the planner are about the planning process itself. Somehow the planner has to interact with the user and the second module takes care of this communication. In section 4.3 the requirements with regard to the interaction with the user (possibly using the PITA) are stated. The last module consists of the data sources KRIS needs to construct routes. The requirements of these sources can be found in section 4.4.

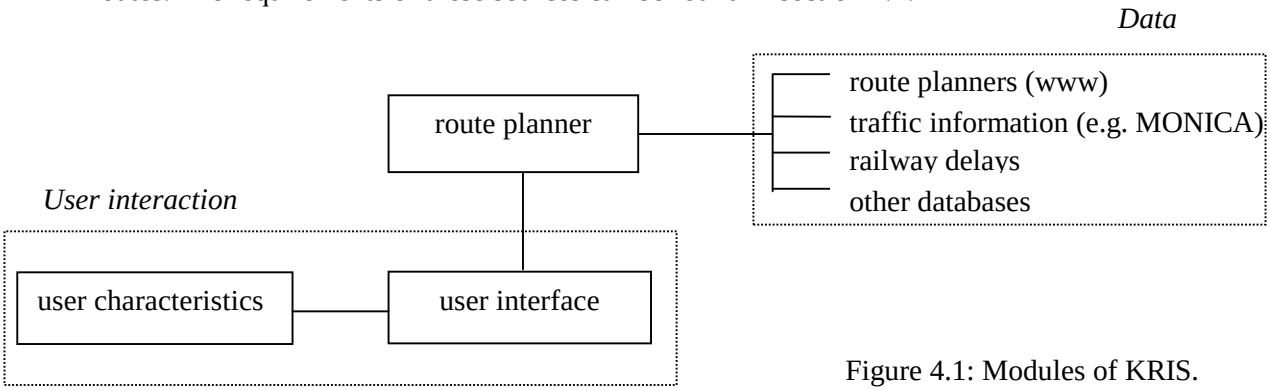


Figure 4.1: Modules of KRIS.

It should be noticed that the design that was made in Figure 4.1 can be compared with the structure of the PITA (Figure 3.1). In Figure 4.2 the part of the architecture of the PITA that corresponds to the three modules that have been identified in this section is illustrated. The labels ‘user interaction’ and ‘data’ in Figure 4.2 correspond to the modules in Figure 4.1. In the PITA the user interaction is done by the agent that also keeps track of the user characteristics by using user profiles. The data sources that are used by the PITA are the same data sources KRIS needs to consult.

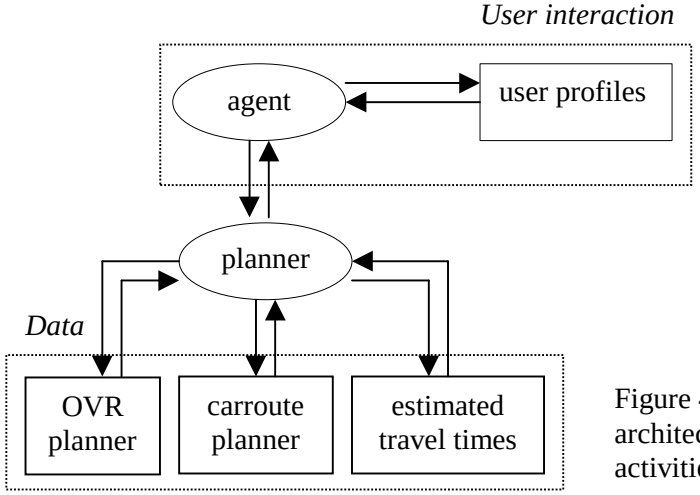


Figure 4.2: Simplified PITA architecture showing planning activities.

4.2 Route planner

As with all route planners, of course the most important requirement is that the 'best' route possible, given a certain situation, is found. A 'best' route can be defined as the shortest route in travel time, the route with the least possible modality changes, the route having the lowest travel cost, or by another qualitative or quantitative measure. Possible measures are stated in section 4.3. In this section the word 'best' will be used as 'best' with respect to one of these measures.
Another important requirement is often contradictory to the 'best route' requirement: this 'best' route should be found as quickly as possible. Unfortunately the time needed to compute this best route increases, when the search space grows. Since the search space of KRIS is very large and complex, a trade off has to be made between the quality of the route found and the time needed to compute this route. Algorithms can be developed to reduce the required computation time and still (almost always) return the best route. Since the computation time can only be reduced by cutting the search space, this implicates not all alternative routes are taken into account, so no guarantees can be made about the quality of the solution found.
For KRIS a requirement has been stated, that incorporates both before mentioned requirements: the incremental requirement. This requirement demands that the quality of the route is improved incremental. As fast as possible a route should be constructed that is a solution, but about which quality nothing can be guaranteed. Now several iterations are done, in which better solutions become available: after the last iteration it is guaranteed the best solution has been found. When the time requirement is important (someone is close to a junction and has to choose), at a given point the search process can be stopped and the route of the last completed iteration is given.
Besides these main requirements, other requirements can be stated. All the requirements of KRIS are stated below:

- *Incremental searching:* as stated before, the route should be constructed by using an incremental algorithm, so always a result can be returned.
- *Optimal route for an individual:* when considering a large number of travellers, it might be a goal to minimise the total travel time of all these travellers. As a result, when there is a traffic jam, not all travellers would be advised a route along an alternative route, since then this route will become congested. Consequently, to minimise the total travel time, some individuals are advised to go along the route that is congested, to accomplish an optimal spread over both routes. KRIS will *not* try to optimise the total travel time of a group of travellers, but will only look at the best route for each individual. Also no capacity restrictions or buffers are taken into account to prevent congestion along routes that are not congested yet, but might become congested when all travellers are routed along this route.
- *Multi modal routing:* KRIS should return multi modal routes (using different types of transportation) when optimal.
- *Dynamic routing:* KRIS should use available congestion data of the highways and delay information of the public transport to construct the best route possible.
- *Reliability*: the same route request (also done at the same time!) should result in the same advised route. Secondly, KRIS should be available 24 hours a day.
- *Modularity:* a lot of different information sources can be used. It is important the working of the algorithm does not depend on individual information sources. It should not matter which information source is used, so in case a better information source becomes available changing to this source can be done easily. This is a lot easier to do when the system has a modular architecture. For example different car planners are available and the working of the system

should not depend on the results of one specific car planner. Different interfaces to different information sources should be developed to guarantee modularity.
- *Scalability:* the size of the underlying network should not matter. The architecture and algorithms should be designed in such a way that a small network can easily be expanded into a huge network.
- *Expandability*: it should be easy to expand the system, to incorporate new information or new transport modalities. For example this is important in case new public transport types become available, or when new roads are opened.
- *Understanding of the knowledge representation*: since it should be easy to expand the system, it is important the knowledge representation is easy to understand. Otherwise it would be a very time consuming task to alter the knowledge base of the system.

Beside these main requirements, other non-essential requirements can be stated, that can improve the system's functionality:

- *Communication*: since one of the most promising applications of KRIS is in combination with a travel assistant, for example the PITA (see chapter 3), the way the system communicates with external agents or programs should be made very clear.
- *Learning*: the system would be more flexible if it could learn from past experiences. If it planned a route, which was not such a good alternative route after all (missed connection for example) and this happens a second time, it would be useful if the system decides to give this route a lower priority in a next advice.

4.3 User interaction requirements

In the previous section the requirements of the planner itself were stated. In this section, the requirements for the interaction with the user are stated: what functionality should be provided. As was stated before an intelligent agent carries out the user interaction when KRIS is used in a travel assistant, like the PITA. In this case the agent acts as a user. On the other hand it should also be possible to use KRIS separately from a travel assistant. Then the traveler himself is the user.

- *Best route definition*: users should be able to choose their definition of a 'best' route. As was stated in the previous section, a best route can be defined in many ways. The user has to be able to choose what is a best route for him: the shortest in time or distance, the cheapest in travel cost, the one having the least modality changes, the most 'secure' one (for example the one with the highest chance of no delays)
- *Personalised routing*: user preferences should be taken into account. These preferences might be stated explicitly by the user or might be observed from the behaviour of the traveller in the past. Examples of personal settings are driver characteristics (the driving style will influence the travel time significantly), the vehicle that is used (a Porsche will be much faster on a certain trajectory than a truck), walking speed (when walking from a car to the train, or from platform to platform), preference for public transport or car (if the traveller wants to read, he may be much more comfortable travelling by train).
- *Graphical user interface:* when used separately from a travel assistant a graphical user interface (GUI) should be available for use at a computer or with a WAP telephone.

- *Explanation facilities:* when a traveller has doubts about a route KRIS advised, or when he just wants to know why a route is faster, it should be possible to get an explanation of KRIS.

4.4 Data requirements

So far nothing was stated about the data KRIS uses and in which way these data should be stored. These requirements are stated in this section:

- *Existing planners should be used*: to avoid a lot of development time is put into the development of already existing software, already existing planners should be used as much as possible.
- *Historical data*: historical traffic data should be used in forecasting travel times when a route is not planned at the moment of departure. Also historical data about delayed public transport vehicles (trains, busses, trams, etc.) should be used as much as possible to be able to forecast the travel time as good as possible.
- *Real-time data*: real-time data, e.g. from the MONICA system (see chapter 8), should be used whenever possible, to have up-to-date information about the state of the highways and public transport network.

Chapter 5: Design

In this chapter the algorithm that was constructed to find dynamic multi modal routes is presented. But first, after a short introduction in section 5.1, a theoretical definition of the problem is given in section 5.2. A way to combine the public transport network with the road network using three-dimensional graphs is introduced in this section. After this theory the algorithm is introduced in section 5.3. In section 5.4 this algorithm is reviewed in the light of the theoretical problem definition that was presented in section 5.2, while in section 5.5 an alternative approach is described. The chapter ends with an illustration of the algorithm: in section 5.6 an example is given by finding the optimal route between Amerongen and Amsterdam.

5.1 Introduction

As was stated in the introduction of this thesis, one of the goals of this graduation work is the design and implementation of a prototype capable of searching the best route from a departure address to a destination address (possibly via another address), using different modalities if desirable and taking all available congestion (road) and delay (public transport) information into account. In the previous chapter the requirements of KRIS were stated. The algorithm that is presented in this chapter has been designed to meet these requirements.
To meet these requirements a lot of data has to be processed: geographical information about the road, railway, bus, metro and tram network, information about the timetables of the different public transport companies, congestion and delay information, etc.. On the other hand, a lot of information providers are available on the Internet. As was stated in chapter 2, route planners for public transport and car planners are available. But these planners are restricted to one modality and/or handle only static, i.e. time independent information. To reduce the search tasks of KRIS the possibilities of incorporating these search engines into the route finding algorithm were investigated.
Summarising this section, an efficient algorithm to find (multi modal) routes had to be constructed, which incorporates other search engines if useful (e.g. efficient).

5.2 Theoretical problem definition

The most straightforward approach to model the problem in a way that it could be solved using mathematical algorithms would be to construct a graph. As with other routing problems, Dijkstra's shortest path algorithm could be used to find the shortest path between origin and destination in this graph. However, in the case of multi modal and dynamic route planning representing the problem by a graph is not as straightforward as with a 'normal' routing problem, when the nodes represent the cities (or junctions) and the links represent the travel times along the roads between them. An example of such a graph is given in Figure 5.1.
When trying to represent the dynamic and multi modal route planning problem in such a way, two major issues have to be dealt with. Firstly, the dynamic aspect of the data has to be taken into account. The travel times between the different edges (cities, stations, and junctions) change in time, and somehow these changes have to be taken into account and incorporated in the graph. When, for example, travelling from Amsterdam to Delft in the morning rush hour, a departure of only 5 minutes later, can affect the travel time by more then 20 minutes, since major congestion may have occurred along the route during these 5 minutes. For example an accident might have happened or a sudden peak in cars that want to access the highway may have occurred.
The second issue that has to be dealt with is the fact that different modalities have to be modelled in the same graph. Train, bus, metro, tram and car trajectories have to be modelled and each of them has own properties that can be different. For example, public transport

vehicles only depart at certain times, while a car can leave at any moment. In this section solutions to deal with these issues are discussed. In subsection 5.2.1 the way dynamic travel times are represented is introduced. Secondly, in section 5.2.2, the representation of different kinds of transportation in a graph is given.

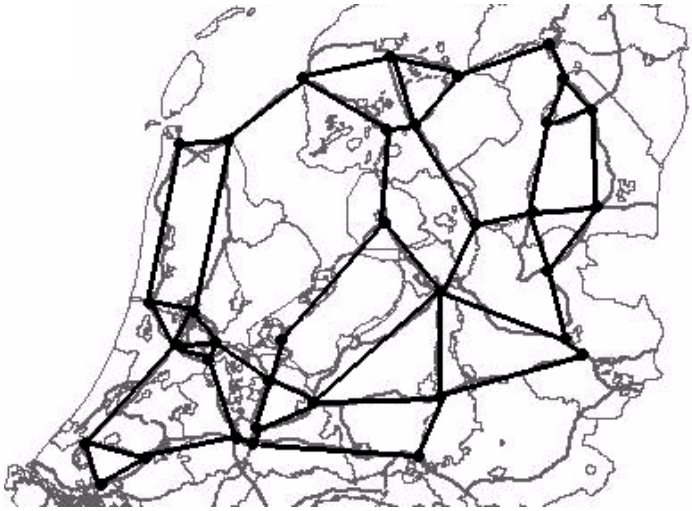


Figure 5.1: Graph representation of the road network that is being considered in this thesis.

5.2.1 Representing dynamic travel times

As was stated in the previous section in some way the time dependent aspect of the travel time has to be taken into account: at different times of the day the travel time between the junctions may vary. A space time extended network (STEN) explicitly represents time by having a complete layer of all nodes of the physical network per time period.
The first occurrence of STEN in the literature can be found in [ford62]. In that paper a time expanded network was constructed to solve a dynamic flow problem: to find the optimal flow in a network with a source and sink, where each arc not only has a capacity but also a traversal time. The problem was stated informally as follows. Given a network G = [N; E] of N nodes and E edges, with source s_s and sink s_d, suppose that each arc of G has not only a capacity, but a traversal time as well. If at each node of G, the commodity can either be transhipped immediately or held over for later shipment, determine the maximal amount of commodity flow from source to sink in a specified number of time periods. An example can be found in Figure 5.2, where the first number on an arc is its capacity in terms of commodity flow per unit time and the second number is the arc traversal time.

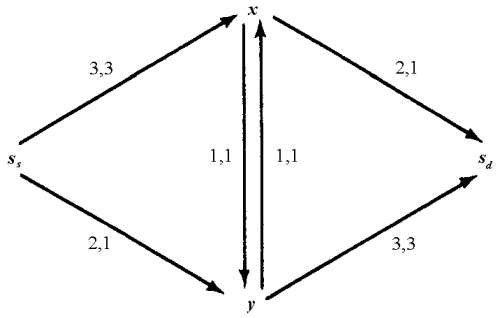


Figure 5.2: Graph representation of the flow through a network. The second number on the links is the traversal time.

To calculate how many units of the commodity can reach t from s in 5 time periods, a time-expanded network version G(p) was constructed from G as follows for p periods of time:

- Corresponding to node x of G, G(p) has p+1 nodes x(t), t = 0, 1, … , p,
- corresponding to arc (x,y) of G, G(p) has arcs [x(t), y(t + a(x,y))] for $0 \leq t \leq p - a(x,y)$, with a(x,y) the distance between x and y,
- to represent hold-overs at node x create arcs [x(t), x(t+1)] for $0 \leq t \leq p - 1$,
- a replica [x(t), y(t + a(x,y))] has capacity c(x,y), whereas the hold-over arcs have infinite capacity.

Figure 5.3 shows the 5 period time expanded network of the network in Figure 5.2. As is proved in [ford62] the maximal dynamic flow problem can always be solved as a maximal static flow problem by expanding the network in the fashion described.

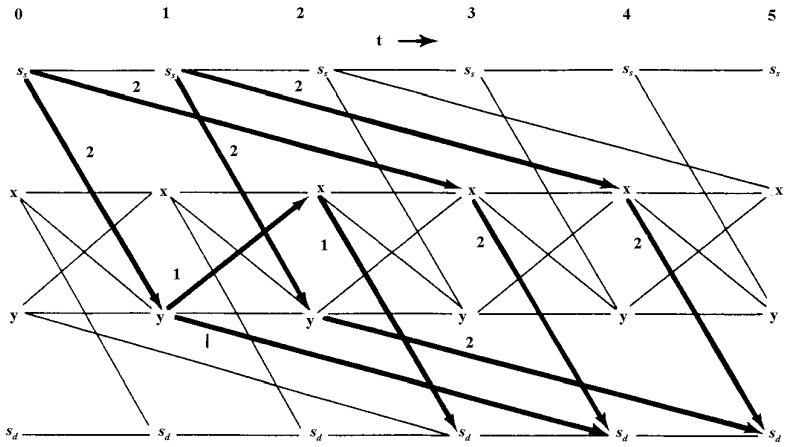


Figure 5.3: Time expanded network of Figure 5.2.

Other applications of space-time expanded networks can be found in [ran96], [chen99] and [evan92]. In all these publications time expanded networks were used to solve traffic assignment models, which are also dynamic flow problems. In KRIS only the shortest travel time has to be computed; no flows have to be optimised (see optimal route for an individual requirement in chapter 4). Consequently, in KRIS the approach in [ford62] can be used, without flow constraints (or a flow of 1 through all edges). Analogue to the construction in [ford62] the dynamic travel times can be represented in a graph that is constructed as follows:

- For each period p create a complete layer of all nodes of the physical network,
- for each node in period p (all nodes with the same t), create links to the nodes it is connected with in the physical network in the corresponding period 'layer' p + d, with d the travel time when starting at period p,
- for each node in period p create a link to the same node in period p + 1 (it is also possible to stop in a node).

An example of a network constructed this way is shown in Figure 5.4. The original graph consisting of nodes A to G is repeated for each time interval. The edges between the nodes A to G (the lowest graph at t = 10:01) represent which nodes are connected to each other. For clarity these edges have been kept in the different layers of the graph to show these connections. The dotted links that intersect the different layers are the actual road connections. Their length (and thus the layer to which they go) represents the travel time when starting at the time of the layer in which they start. For each layer and for all nodes, outgoing links are constructed according to the travel time at that moment. It should be

noticed that in Figure 5.4 not all the links are shown, since that would have resulted in a cluttered figure.

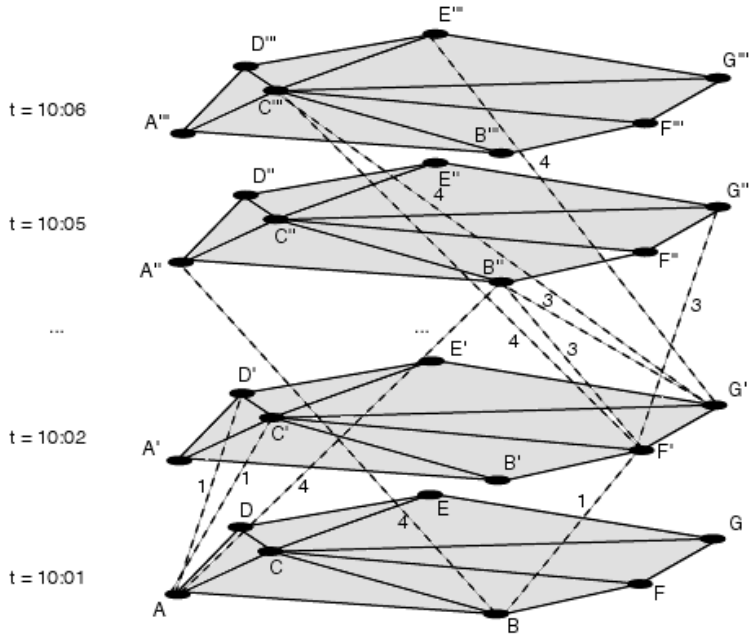


Figure 5.4: A three dimensional graph.

This approach places one constraint: the travel time has to be discretized to intervals. When using a very high sample rate an enormous graph is required, while a lower sample rate results in loss of information. For example, when a sample rate of one minute is chosen and the travel time from A to B is 22,4 minutes and the travel time from B to C 12,4 minutes, the discretized travel times in the graph will be 22 and 12 minutes. The total travel time will become 24 minutes, while actually this would be 24,8 minutes, so information is lost.

5.2.2 Representing different modalities

As mentioned in the introduction of this section, a solution has to be found to represent different kinds of modalities, since each modality has its own properties. Only at certain transfer points one can change modality. As a consequence each modality has its own (partial) graph, which intersect at these transfer points. An example is given in Figure 5.5, where the intersections between the graphs of two modalities are shown by connecting links. The lower layer consists of a graph representation of (for example) the road network, the upper layer (for example) of a representation of the railway network. The links between the different layers represent the fact one can change modality at the connected junctions. In this example (of a road and a railway network) the nodes that are connected are stations. The links between the layers represent the time needed to change modality. In this example this can be the time needed to walk from the parking place to the station itself.
The example in Figure 5.5 connects (only) two-dimensional graphs for clarity. In reality the links connecting different modalities will be between the 3-dimensional graphs of different modalities. In this graph the links will also represent the time needed to change modality. In each layer links will start to the corresponding layer representing this travel time. Since showing all the connections in this graph would result in a screen full of lines the intersections were only illustrated between two two-dimensional graphs (Figure 5.5).
One can imagine that when all different kinds of modality that can be used are incorporated, this conjugated graph will become very large: all tram, metro, bus, train and road networks and their mutual connections will have to be represented in this graph.

The major difference between the different kinds of modalities is the fact that some modalities are bound to time tables. Trains, busses, metros and trams only depart at given times, while a car can depart at any time. When the weights of the vertices represent the travel time needed to travel from one junction to another, somehow the fact some modalities are bound to timetables has to be represented. For the roads, the time needed to get from one node to another at a certain time of the day should be the length of each vertex. As was stated in the previous subsection the travel time has to be computed for each period p that is used to construct the space-time expanded network. The time needed to get from one station to another (or from one bus stop to another) at period p (a certain time of the day) can be represented by the travel time between those stops, plus the waiting time until the train, bus, metro or tram leaves. In this way the time tables are incorporated, but consequently the travel times plus waiting time have to be computed for each interval.

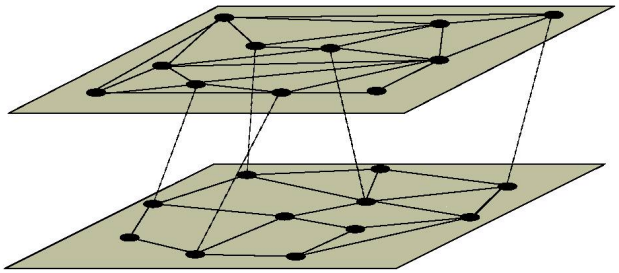


Figure 5.5: Two connected graphs of two networks of different modalities.

5.3 Algorithm

In the previous section the problem definition was given by representing the dynamic aspects and different modalities into a three-dimensional graph which contains all (sub) networks of the different modalities. These (sub) networks are connected at the transfer points. A solution can be found by finding the shortest path in this graph. An algorithm should be constructed to find the shortest route in this three-dimensional graph, like Dijkstra's algorithm for a 'normal' two-dimensional graph. As was stated in the introduction of this thesis probably the performance of such an algorithm may be very poor because of the size of the graph that has to be constructed.
To proof this statement such an algorithm was constructed. In chapter 7 this algorithm is introduced, since it is only used to find a partial solution: the best route in the road network. In chapter 10 a comparison is made between this brute force search approach and an expert system approach, which is introduced in chapter 6. Although it would be very straightforward to use the Dijkstra algorithm to find a complete solution (the same algorithm could be applied to the graph that was constructed in the previous section), this was not feasible because of the following points.
Firstly, constructing a Geographical Information System (GIS) consisting of all roads and train, tram, metro and bus connections would have taken far more time than available for this graduation work. Secondly, the requirement of modularity implies different modules should be used for the different tasks of route planning. Thirdly, the construction of the graph for each minute of the day would be a very time consuming process, slowing down the computation time considerably. Finally, it would be a waste of time to start developing a complete planner, since very good static planners have been developed, whose results can be used very well to incorporate the dynamic information. To use a Dutch saying, a lot of work would have been done twice.
Because of these drawbacks another algorithm was constructed, using a modular approach. In Figure 5.6 a schematic overview of the constructed algorithm is given. In the boxes the different routing activities are shown, while the arrows denote the routes that are passed from

one activity to another. In the following subsections the different steps of the algorithm are discussed.
In section 5.4 the algorithm is reviewed with respect to the theoretical problem definition, which was presented in section 5.2, while in the next section another approach to find a solution is presented. This chapter ends with an example of the application of the algorithm (section 5.6.).
In the discussion of the algorithm often is being referred to the fastest route. As was stated in the previous chapter the label 'best route' can also be based on another objective then fastest route. In the case the user prefers another objective then fastest route to compare different routes, KRIS should make comparisons according to these preferences. In stead of computing the shortest travel time the costs or the number of transfers of each route should be computed for example. To prevent the description of the algorithm to be filled with possible objectives, the objective shortest travel time will be used throughout the description to illustrate the working of the algorithm. Consequently this objective could be replaced by other objectives.

5.3.1 Static car and public transport routes

The first steps of the algorithm are to construct static car- and public transport routes (steps 1 and 2 in Figure 5.6). According to the modular design requirement no specific route planners have to be used. For example, planners available on the Internet can be used. Several planners are available for planning a car route, while only one planner is available to find the best public transport route (http://www.9292ov.nl). The only restriction on the different route planners is placed by the need to translate the routes to a format that can be used by the other modules of KRIS. For planning the car route the Ilse route planner is used (http://kaart.ilse.nl), although, according to the modularity requirement, other planners could be used as well.

5.3.2 Adjusting the static routes to dynamic information

When the static routes are found, they are adjusted to the available information about congested roads, railroad / road work, closed roads, delayed or not riding trains, etc. (step 3). This task is carried out by different modules for each type of transportation as far as dynamic data are available. The car travel times can be updated by using travel time estimations produced by MONICA, the monitoring system that is placed along all major highways in the Netherlands. All trajectories of the static route that was produced in the previous step can be updated according to the estimations for that time of the day. In chapter 8 the MONICA system and methods to estimate travel times are introduced. The travel times of the public transport can be updated according to known delays and estimations about the effect of these delays, also as far as data is available.
Now realistic travel times of the available routes are available, the actual dynamic route planning can be started (steps 3a and 3b). As was stated in the introduction of this thesis, two implementations of the dynamic route planning for the car have been made. The first one is based on an expert system approach, while the second one applies a shortest path algorithm in a graph consisting of the freeways of the Netherlands. These approaches are discussed in chapters 6 and 7. No implementation of the module that carries out the dynamic planning of the public transport route has been made yet.
The results of steps 3a and 3b are optimal car and public transport routes. Since important design issues of these different approaches are discussed in the following chapters, they are not covered here.

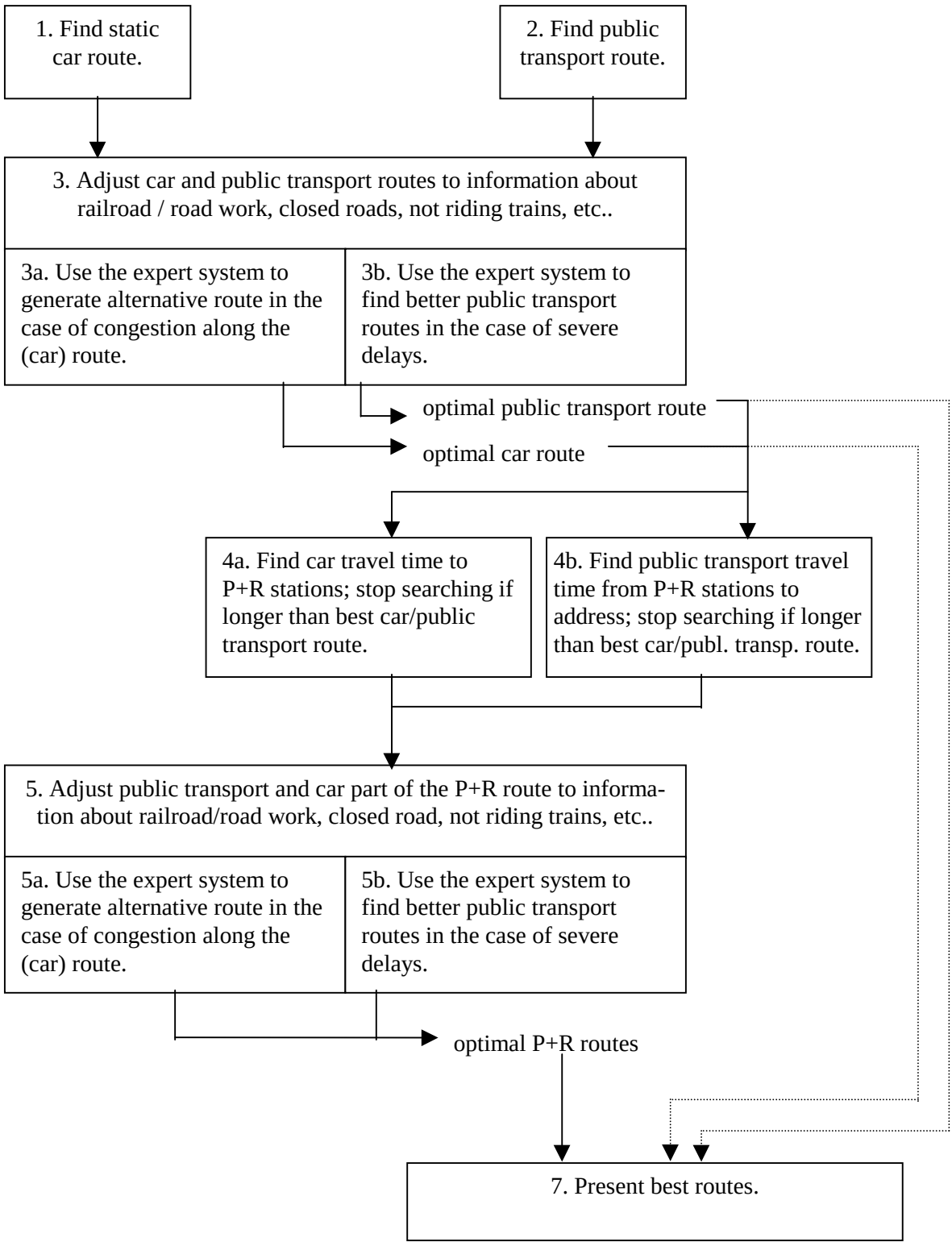


Figure 5.6: Flow chart of the algorithm to find the optimal route.

5.3.3 Finding multi modal routes

When the optimal public transport and car route have been found, the next step of the algorithm is to search for P+R (park and ride) routes. In Figure 5.6, this process is divided in steps 4a to 5b. To find the best multi modal route, the fact that there are only a limited number of P+R stations in the Netherlands is used. Since at stations that do not have P+R facilities parking is virtual impossible, these stations cannot be used in a multi modal route that uses a car to get to the station. This is a very important property, since it reduces the search space significantly. Instead of all stations in the Netherlands (approx. 300), only 30 have to be examined. Consequently, a list of all P+R stations in the Netherlands is used to find the best P+R route(s). When new stations become available it is easy to expand this list, according to the expandability requirement (see chapter 4).
In the remainder of this section firstly the algorithm that was constructed to find P+R routes will be properly defined. Then a textual explanation will be given.

A brute force approach to find the best P+R route can be defined as follows. Assume that there are N P+R transfer points. Assume that we want to travel from A to B, passing a P+R station. We travel by car to a P+R station and then continue by public transport (see Figure 5.7). To include the routes from A to B that only travel by car or public transport we assume that A and B are transfer points too.

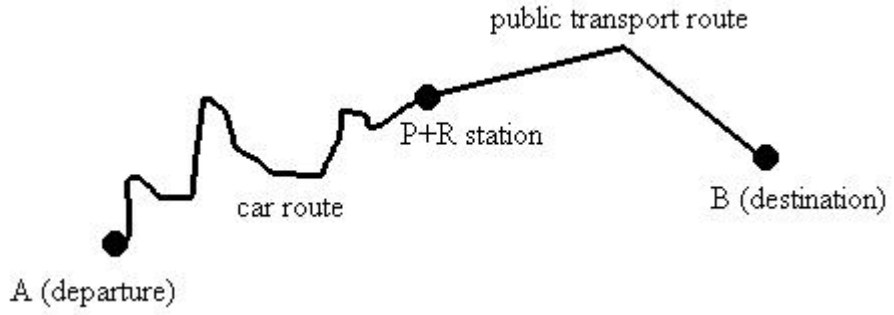


Figure 5.7: Schematic overview of a P+R route from A to B.

To find the shortest route:

a. Calculate $D(P+R)_{best}$, the travel time of the shortest route from A to B while only using one modality (car or public transport).
b. Calculate $S(P+R)_i$, the travel time of a static route from A to B, composed of the optimal static car route from A to $P+R_i$ ($S(P+R)_{i,car}$) and the optimal static public transport route from $P+R_i$ to B ($S(P+R)_{i,public\ transport}$).
c. Order $S(P+R)_i$, starting with the shortest P+R route $S(P+R)_i$ for the P+R routes that are shorter then $D(P+R)_{best}$.
d. Starting with the shortest P+R route $S(P+R)_i$ transform the routes to dynamic routes $D(P+R)_i$. If $D(P+R)_i$ is shorter then $D(P+R)_{best}$ this route will become $D(P+R)_{best}$.
e. If $D(P+R)_{best}$ is shorter then $S(P+R)_j$ then stop transforming static routes into dynamic routes: the best route is found.

First the (car) travel times from the departure address to all the P+R stations that are on this list are computed ($S(P+R)_{i,car}$). These travel times are then compared to the best route found so far (the fastest route of the public transport and car routes that were found in step 3, $D(P+R)_{best}$). P+R stations that require more travel time to be reached then the travel time of this best route are removed from the list, since these stations will never be in a route that is faster then the best route found so far. Subsequently the same process is carried out for all routes from the (remaining) P+R stations to the destination ($S(P+R)_{i,public\ transport}$). Using the

static public transport planner the travel times for these routes are computed and the stations that are too far away from the destination address are removed from the list. Finally the travel times of the car and public transport route of each P+R station that is still on the list are added ($S(P+R)_i$) and again those stations are removed that will not be in a faster route then the best one found so far. Implicitly a small threshold is used in this process, since only the 'raw' travel times op both routes are added. When constructing the final route, a few minutes have to be added to park the car at the station and some extra time has to be added to the car route, to allow slight delays.
For the 'remaining' P+R stations (the P+R stations that are in a route of which the static travel time - or other objective - is less then the best route found so far) the same process as described in subsection 5.3.2 is carried out: the routes $S(P+R)_i$ that are shorter then $D(P+R)_{best}$ are now reviewed using dynamic data, starting with the route that has the shortest $S(P+R)_i$. In the previously given description of the algorithm this is described in step d. First the found static route is updated according to the available information (step 5). If significant delays are found, the expert system (or the graph algorithm) is used, to generate the best possible route to the station (step 5a). Now the arrival time at the P+R station is known, so the best possible public transport route from the station to the destination can be constructed. Again, if there are any significant delays the expert system is used to find better routes (step 5b). A very big advantage of this approach is that the same expert systems that were used to carry out steps 3, 3a and 3b can be used again. As was stated in the previous subsection, detailed information about these steps can be found in chapters 6, 7 and 8.
It may be clear that the different steps to construct the best P+R routes are all aimed at a reduction of the search space. Since the computation time of steps 5, 5a and 5b of the algorithm determines the largest part of the total computation time, these steps have to be carried out as little as possible, according to the minimal computation time requirement (see chapter 4). Of course it would have been possible to determine the routes from departure to destination address for all P+R stations and picking the fastest route using these steps, but this would be very inefficient because of the fact a best travel time is already available (the result of step 3). This travel time can be used to cut the search space: P+R routes that are slower at one of the two parts (car or public transport) will never be faster over both parts. P+R routes that have a longer travel time when the travel times of the two static routes are added, will definitely not be faster if dynamic information is added (step 5), since this will only delay a route.
It should be noticed that the described (and used) algorithm is a brute force approach. It is also possible to apply an artificial intelligence approach in deciding which P+R stations are being reviewed. For example heuristics can be used about which stations are often used in P+R trips, the route to which stations is almost always congested, etc..

5.3.4 Comparison and presentation

The last step of the algorithm consists of a comparison between the different alternative routes and the presentation to the traveller. This is a very straightforward process, although adaptive learning mechanism could be used, to be able to make the comparison between the different alternative routes using the preferences of the traveller. In [vark00] and [rog99] the possibilities of such adaptive mechanisms are described in detail. They are outside the scope of this thesis.

5.4 Reviewing the algorithm

When reviewing the algorithm with respect to the theoretical framework that was given in section 5.2 the following remarks can be made. Firstly, it can be stated that the graph is split up into different sub graphs for the different kinds of modalities: optimal routes are found in the sub-graph of the road network (step 1) and in the sub graph consisting of all public transport modalities (step 2). These sub-graphs were illustrated in Figure 5.5. In this way, the

modularity requirement is met and the problem of representing the time tables in a graph that also consists of 'normal travel times' has not to be solved.
Secondly, it should be noticed that first routes are found in a 2-dimensional graph (step 1 and 2), without using dynamic travel information. As a result, a solution, which may not be the best one, is found very fast, according to the incremental requirement. In step 3, for both sub-graphs, the third dimension of time is added. This step can be carried out in different ways. An extensive shortest path algorithm can be applied, as is described in chapter 7. This is an approach similar to Dijkstra's shortest path algorithm. On the other hand, it is also possible, to use a knowledge based approach and to search in an 'intelligent' way, as is described in chapter 6. In Figure 5.8 an illustration is given in how the third dimension of time is added without having to construct the complete three-dimensional graph that was introduced in section 5.2.1 (see also Figure 5.4). Only the travel times at relevant times are computed, which can be seen as some kind of gliding table of travel times for different departure times for each link.

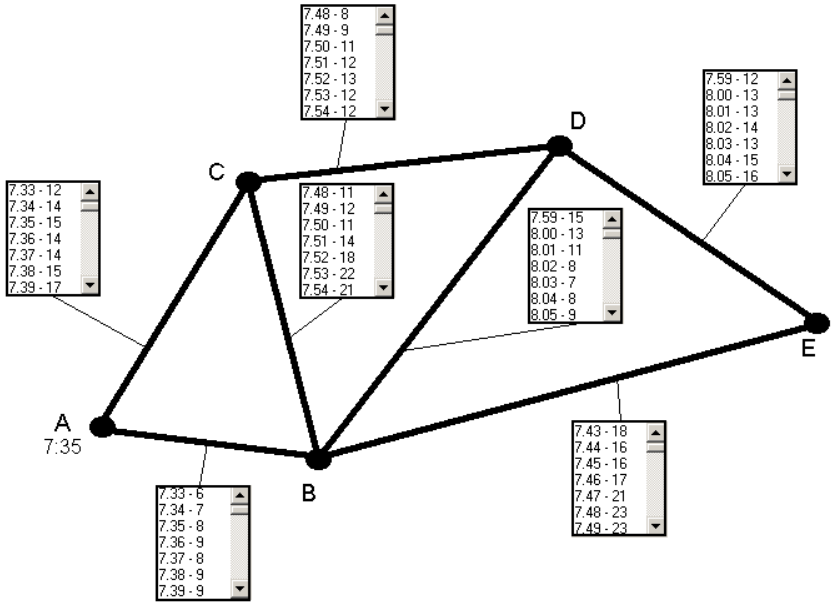


Figure 5.8: Gliding travel time estimates for a route between node A and node E.

Thirdly can be remarked that in steps 4 and 5, a solution in the complete (combined) graph is searched. Searching optimal routes in both sub-graphs to all transfer points does this. Firstly, in the car route graph the best routes from the departure address to all transfer points are found (since the car can only be used in the first part). Then the best routes from all transfer points to the destination address are constructed. These routes are combined in all possible ways and the best one is chosen. Considering the starting time of 7:35 am, for each link the relevant travel times are calculated. In the list-boxes connected to each link the travel times for each departure time of the starting node of the link are shown.

5.5 Alternative approach

An alternative approach would have been to start with the public transport route from the departure address to the destination, which was a result of step 3. Beginning with the first P+R station that is reached, the car is substituted and the total travel time is computed. This is done for all P+R stations along the route. This is a very simple approach that does not require too much computation time, since only one public transport route is taken and only a few P+R stations are investigated. On the other hand, this approach is only a sub-optimisation, since completely other routes, along P+R stations that are not along the optimal public transport route are not investigated. The minimal computation requirement is met very well at the cost

of the optimal route requirement. In the case of KRIS the trade off between the quality of the found route and the computation time needed to find the route was made in the favour of the quality of the route, since in this case there quite a reasonable chance a route that is (a lot) faster is not taken into account.

5.6 Example

The way KRIS finds the best route can best be illustrated using an example. This will be done in this section. The different steps of the algorithm will be carried out to find the optimal route for a traveller, who has to go from Amerongen to Amsterdam at a Tuesday morning. He would like to leave at 8 a.m. and has a car available for the trip. But he is afraid there will be a lot of congestion along the A12 and A2 and (as a result) is interested in the possibilities of travelling by public transport. The departure address is Overstraat 1 and the destination address Omval 315. In Figure 5.9 both places are illustrated on a map.

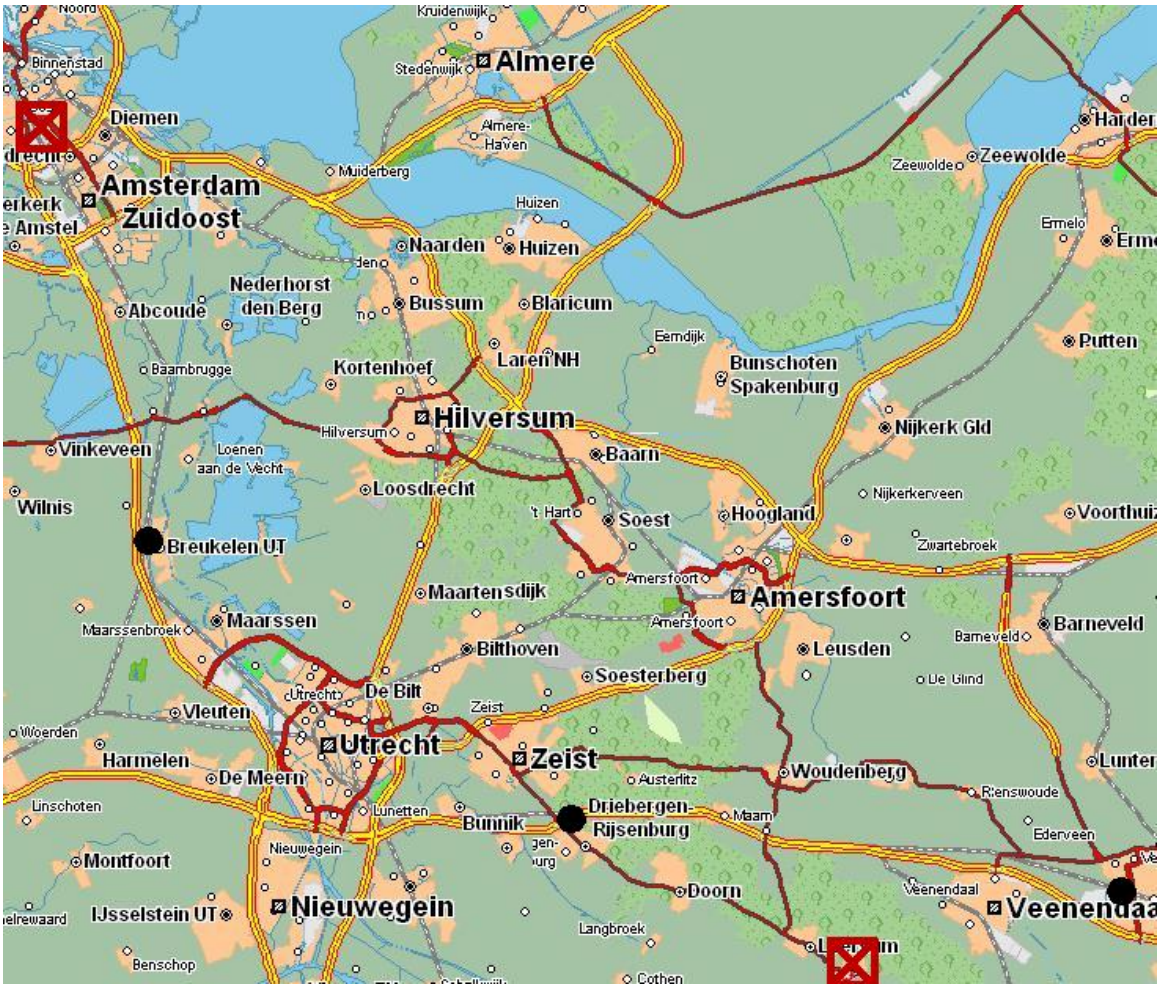


Figure 5.9: Map of the highways and railways between Amerongen and Amsterdam. The departure and destination address are denoted with the red cross marks, while the P+R stations that are being considered are indicated with black dots.

5.6.1 Finding the static routes (steps 1 and 2)

As can be seen in the schematic illustration of the algorithm (Figure 5.6) the first two steps of the algorithm are to find a static car and public transport route. Route planners that can be

found on the Internet can do both these. Of course, also a route planner running on the same machine can be used. As said before, requests are done to the Internet sites http://kaart.ilse.nl and http://www.9292ov.nl. The results of these requests can be found in Tables 5.1 and 5.2.

Table 5.1: Step 1: Find optimal static route by car.

Directions	Minutes
Drive to the A12, ramp Maarsbergen	9
Follow the A12 to the Oudenrijn junction	16
Follow the A2 until it ends	17
Drive along the Amstel to the Berlagebrug and Omval	1

Total travel time: 0h42. Arrival time: 8.42.

Table 5.2: Step 2: Find optimal route using public transport only.

Arrival	Departure	Station / bus stop	Action
	7.42	Overstraat 1	Walk to busstop
7.45	7.47	Bus stop Amerongen	Take bus 51 to Driebergen-Zeist
8.23	8.28	Driebergen Zeist station	Take train to Utrecht CS
8.40	8.46	Utrecht CS	Take train to Amsterdam CS
9.07	9.07	Amsterdam Amstel station	Walk to destination
9.10		Omval 315	

Total travel time: 1h29. Arrival time: 9.10.

5.6.2 Incorporating dynamic travel data (step 3)

It may be clear that the route by bus and train is much slower than the static planned route by car. Since the route is planned in the rush hour, one would suspect severe delays along the A12 and A2, since these are very heavily used freeways in the Netherlands. The investigation of these delays is done by the expert system (or shortest path algorithm) in step 3 of the algorithm. The detailed operation of the expert system and the shortest path algorithm are described in chapters 6 and 7. In Table 5.3 the results of the adjustments to dynamic traffic data are shown. In this case no alternative car routes were found, although the route is delayed significantly.
The results of step 3b, where it is checked if there are any delays on the rail network and if so, other public transport route possibilities are investigated are as follows. At this time there are no railroad works planned and historical data shows the connections are almost always made. So no adjustments are made. The travel time using public transport stays 1h29.

Table 5.3: Dynamic travel time from Amerongen to Amsterdam at 8 am.

Directions	Minutes
Drive to the A12, ramp Maarsbergen	9
Follow the A12 to the Oudenrijn junction	27 (+11)
Follow the A2 until it ends	31 (+14)
Drive along the Amstel to the Berlagebrug and Omval	1

Total travel time: 1h08. Arrival time: 9.08.

5.6.3 Finding possible P+R stations (step 4)

Knowing the optimal car and public transport route the next steps of the algorithm can be applied. For all available P+R stations in the Netherlands the car routes from the departure address to each station and the publics transport routes from these P+R stations to the destination address are computed. When the travel time to or from the station becomes larger then the shortest route found so far, the P+R station is deleted from the 'possible P+R stations list'. For the stations that remain, the car and public transport route times are added, and again

those stations are removed from the list that result in a longer route then the shortest route found so far. The car and public transport routes of the remaining stations are now investigated for delays and for better alternative routes. In the case of this example only the stations that seem to be a reasonable alternative route are investigated, since a discussion of all alternative stations would require far too much space, than is useful to illustrate the working of the algorithm. The stations that are considered in this example are Ede-Wageningen, Driebergen-Zeist and Breukelen. They can also be found on the map in Figure 5.9.
When the car routes to the P+R stations have been made, the travel time of the public transport part of the route has to be estimated. This is an estimation, since the 'real' departure time is not known yet: the car routes still have to be investigated for congestion and as a result can be updated. In spite of the fact that the travel time of the public transport part of the route can not be computed exactly, an estimation is made, to reduce the search space. The estimated travel times are used to remove P+R stations of the list, that are not a reasonable alternative route, in the way that was described before. In Table 5.4 the travel times from the destination to the stations and from the stations to the destination are listed. It should be born in mind that these travel times were also computed for all other P+R stations that are on the P+R list (see appendix C), but that these results are not shown here, since it would have resulted in a very large table.

Table 5.4: Travel times from and to P+R stations.

	Breukelen	Driebergen-Zeist	Ede-Wageningen
Car time	56	20	29
Public transport time	24	37	51
Total time	1h20	57	1h20

5.6.4 Adjusting the car and public transport route to dynamic travel data (step 5)

For the P+R stations that survived the previous phase, now the optimal routes, taking into account the dynamic travel data, have to be found. Since only the route via Driebergen-Zeist is shorter than 1h08, only this route is updated according to the dynamic data. This is the same process as was already illustrated in section 5.6.2. First this step is applied to the different car routes. Now the arrival times at the P+R stations are known. The first possible departure time of the traveller from this station is a few minutes later, since a certain margin has to be applied to the time of arrival at the station and the departure of the train. The traveller has to be able to park his car and of course it is always possible some unforeseen delays occur along the route to the station. Consequently the train departure should be a minimum of 4 minutes later, depending on the user profile of the traveller. Now the departure time of the train is known, the public transport expert system can start finding a route. The resulting route is shown in Table 5.5.

Table 5.5: P+R route via station Driebergen-Zeist.

Arrival	Departure	station / bus stop / freeway	Action
	7.55	Amerongen	Drive to station Driebergen-Zeist
8.15	8.21	Station Driebergen-Zeist	Take train to Utrecht CS
8.30	8.32	Utrecht CS	Take train to Amsterdam CS
8.55	8.56	Amsterdam Amstel station	Walk to destination
8.58		Omval 315	

Total travel time: 1h03. Estimated arrival time: 8.58.

5.6.5 Presentation of the different alternative routes (step 6)

All different possibilities are now presented to the traveller. The travel time by car if there would not have been congestion is given and the estimated travel times if there would have

been congestion, with the suggested best route. Thirdly the travel time and route when using public transport is shown. Finally the P+R routes that are faster then the best travel time of the public transport and car route are given. Also an advice of the system can be given, on the base of the most 'secure' alternative route (the one with the smallest chance of delays), the fastest alternative route, the one with the least switches of transportation modality, the one with the lowest fare costs, etc.. When a traveller uses the system often, knowledge about his preference can be used in the choice of which alternative routes are shown.

5.6.6 KRIS used within PITA

The example of KRIS in the previous sections did not illustrate how KRIS could be used in the PITA. To show the potential of KRIS when used within the PITA the route described in the previous subsections is used in a scenario that is presented in this subsection.
The traveller (still) has to travel from Amerongen to Amsterdam, but thinks the congestion will not be as serious as KRIS forecasted. He takes his car and starts to drive. Fortunately his thoughts were not bad at all, since he is not confronted with any severe congestion along the A12. After he turned onto the A2 it still seems quite quiet along the road. But just a few minutes after he switched to the A2 a serious accident happens close to Amsterdam. Due to this accident only one out of three lanes can be used and serious congestion starts to build up fast. The MONICA monitoring system detects this congestion and the PITA is informed through its change module. Immediately the PITA starts to re-route. KRIS is given a new route request now from the ramp 'Oog in al', close to where the traveller is, to the Omval 315. KRIS starts to plan a new route and finds a new alternative route: the next exit ramp is Breukelen and the station is only an one minute drive away from this ramp. In a few minutes a train to station Amsterdam Amstel will leave so the driver is informed about this possibility, that will bring him within 25 minutes in Amsterdam. Also an estimation of the travel time along the A2 is given: 50 minutes. Although he will have to take the train back to station Breukelen on the way back this time gain is enough for the driver to decide to take the train from Breukelen. Thanks to the PITA his travel time will be reduced by 25 minutes.

Chapter 6: The expert system

In chapter 5 the design of the total system was given: in Figure 5.6 a schematic overview can be found. In section 5.3 it was stated that two implementations were made to find the best (dynamic) car route (steps 4a and 6a in Figure 5.6). In this chapter the expert system approach will be discussed. This will be done in the same way the design of the total system was discussed. First the requirements of the expert system will be stated. These requirements are deduced from the requirements of the total system, which were given in chapter 4. In section 6.2 the problem definition of the expert system is given, followed by a discussion of the design choices that were made. These choices are clarified in section 6.3 to 6.7. Finally, in section 6.8, implementation details are given.

6.1 Requirements

In chapter 4 the requirements of KRIS were given. On the basis of these requirements, the requirements for the expert system can be stated, since the functionality of KRIS is defined partly by the functionality of the expert system. An example is the quality of the car part of the route, which is defined, by the quality of the route found by the expert system. Since all requirements were explained in the previous chapter, in this section only relevant information with respect to the expert system is given with each requirement.

- *Incremental searching:* when the first solution that is better then the found static route is found, this one should be available until a better one is found: until the expert system has finished searching, the best route found so far should be available.
- *Dynamic routing along highways:* the solutions of the expert system should be the best possible with respect to the available dynamic information. Only freeways of which dynamic data are available should be taken into account, since otherwise no guarantees about the solution can be given.
- *Reliability*: the same route request (also done at the same time!) should result in the same route.
- *Expandability*: it should be easy to expand the rule base with new rules, when new roads are opened or when existing freeways (or roads) that are currently not incorporated by the expert system have to be added.
- *Understanding of the knowledge representation*: since it should be easy to expand the expert system, it is important the knowledge representation is easy to understand. Otherwise it would become a very time consuming task to alter the knowledge base of the system.
- *Modularity:* with regard to the previous two requirements, it is important the rule base is modular. A modular rule base is easier to understand, change and expand. Different rule sets for congestion along each highway should be constructed.
- *Best route definition*: the expert system only returns optimal routes with regard to the least travel time needed. To satisfy other optimal route definitions (least costs, least switches, etc.) KRIS should use other search procedures, or other expert systems should be developed.
- *Personal settings*: personal settings that influence the driving speed of the car should be taken into account by the expert system. This is not the task of the expert system itself, but these preferences should be taken into account by the travel time estimation module, which is used by the expert system.
- *Real time*: the expert system should be able to perform in a real-time environment: as soon as a solution is found, although its quality can not be guaranteed yet, it

should become available. This solution should also be available as fast as possible.

- *Historical data & real time data*: the best data possible should be used, but as was mentioned with the previous requirement, the travel time data itself are not of concern of the expert system. A different module has to be made that returns estimated travel times for certain trajectories. This is also discussed in section chapter 8.
- *Existing planners should be used*: the expert system should start reasoning with a static route.
- *Explanation facilities*: it should be possible for a traveller to ask why the expert system advised a certain route.

6.2 Problem definition

Actually, the problem definition is already given in section 5.2. The definition that was given in this section also included different modalities. The expert system will take only one modality, the car, into account. Consequently, the problem definition that was given in section 5.2 can be used, with the restriction that only one modality is used.

6.3 Design

The design of the expert system is discussed on the basis of the 'incremental model', that was proposed by Mateu Tomas Saborido [sab92] to describe the development of an expert system. According to this model, the design phase starts with the process of knowledge acquisition, which is followed by the development of the conceptual model. The results of the design (called system specification in the incremental model) phase are a conceptual model and a so-called design model. The conceptual model is a result of an extensive analysis of the knowledge of the experts and the environment in which the expert system will be used. The design model is a functional and architectural model of the final system. Both models are being introduced in sections 6.6 and 6.7. Firstly, in section 6.4 the domain of the expert system is delimited. This is done according to the problem definition and system requirements that were stated in the previous two sections. This section is followed by an introduction on how the knowledge elicitation process was carried out (section 6.5).

6.4 Domain

As can be seen in Figure 5.6 the expert system gets a static planned route as its input. If there is congestion along this static route, the expert system should start reasoning and try to find a better route. The domain of the expert system determines in which search space the expert system should search. As a consequence the quality of the found shortest route depends mostly on the domain that is chosen. The search domain that was chosen consists of the part of the freeway net (in the Netherlands) on which the actual speeds, density, etc., are monitored. This network is shown in Figure 6.1. Only this part of the road network in the Netherlands is taken into account because of the requirements that were stated in section 6.1. The requirement *dynamic routing along highways* implicates route planning should only be done along freeways of which dynamic data is available. Secondly, the *historical data & real time data* requirement requires historical and real time data should be available, which is only available along this part of the road network. Of course, it should be easy to expand the rules of the expert system, when dynamic data of more roads become available. This is guaranteed by the *expandability* requirement.

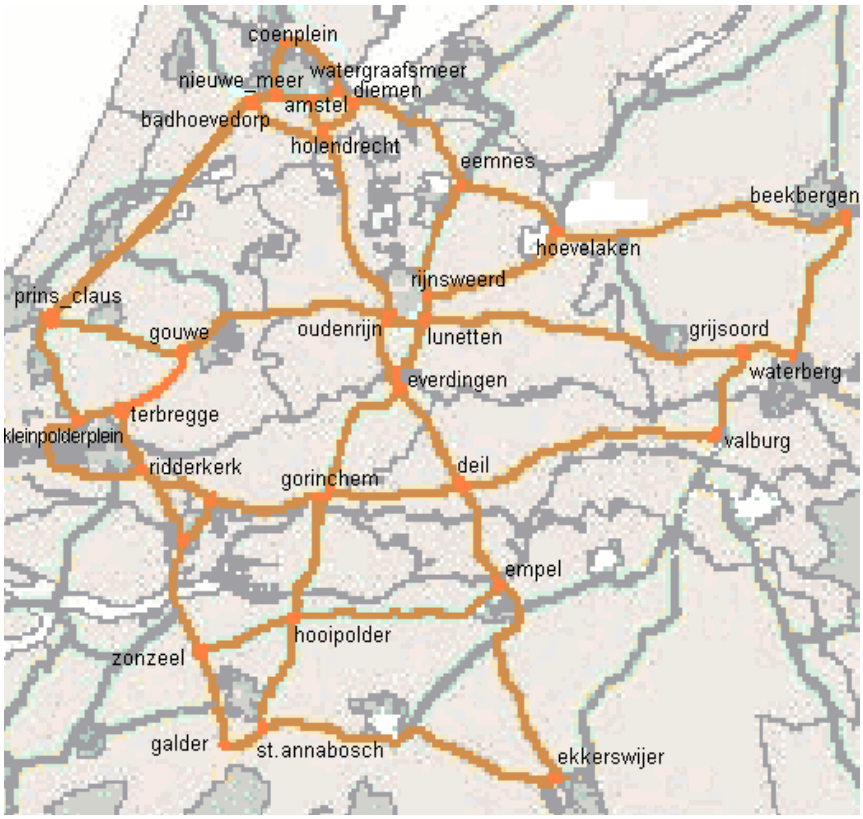


Figure 6.1: The freeway network that is considered by the expert system.

Since the search domain is restricted to the freeway net, found solutions can be sub-optimal. Only the part of the (static) route along the freeway net that is monitored will be optimised. If the static route is not along a road in the freeway net, even no alternative route route is found. When the static route uses one or more roads of the freeway net that belong to the search domain it is possible sub-optimal routes are found. This is illustrated in Figures 6.2 and 6.3 and the corresponding example, where the route is only optimised for the part that is along the freeway net. Consequently, the parts of the route to the first freeway that is in this network and from the last freeway that is in the network are not changed at all. It may be possible that it is faster to travel along roads that do not belong to the network to another road in the network and to travel a completely different route. So the routes that are found by the expert system may be sub-optimal: they are only optimal from the freeway network point of view, not from the total road network point of view.

Figure 6.2: Fastest route from Amsterdam to Delft

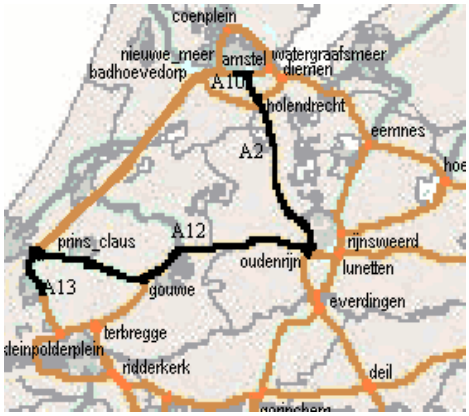


Figure 6.3: Route from Amsterdam to Delft when the A4 is congested.

The shortest route from the Omval in Amsterdam to the Julianalaan in Delft would be along the A10, A4 and A13. The ramp 'RAI' would be used to get onto the A10, while the ramp 'Delft' would be used to turn off the A13. The route from the Omval to the ramp 'RAI' is illustrated in Figure 6.4, while the route from the ramp 'RAI' to the ramp 'Delft'is illustrated in Figure 6.2. When severe congestion occurs along the A4, the expert system would generate an alternative route along the A2 and A12, which is illustrated in Figure 6.3. From ramp 'RAI' the route leads to the A2, until the 'Oudenrijn' junction, where the route continues along the A12. At the 'Prins claus' junction one changes to the A13 again. This route is the optimal route from the ramp 'RAI' to the ramp 'Delft', but not the optimal route from the Omval to the Julianalaan. It would be faster, to drive from the Omval to the A2 immediately, in stead of first driving to the 'RAI' ramp and then driving to the A2. This route is illustrated in Figure 6.5. As can be seen in Figure 6.4 and 6.5, it is faster to use the route in Figure 6.5 to get onto the A2, then to travel to the 'RAI' ramp first and to travel to the A2 along the A10 afterwards.

Figure 6.4: The route to the 'RAI' ramp (A10).

Figure 6.5: The route to the 'Amstel' ramp (A2).

Concluding, the domain can be summarised by the following points:

- Only highways of which dynamic data is available are taken into account (Figure 6.1),
- dynamic and historical data are used to find the best route,
- the input consists of a start and finish ramp and a route between these ramps,
- optimal routes are searched between a start and finish ramp only at the monitored highways.

6.5 Knowledge acquisition

The main part of knowledge of the expert system consists of alternative routes in the case of congestion along a part of a road. Consequently, for each part of a highway that can be congested the alternative routes should be known. This knowledge can be made explicit in two ways. Firstly, experts can be interviewed. These experts should be experienced 'traffic jam travelers' that often have tried alternative routes in the case of congestion along a part of the freeway they normally use.
Secondly, the map of the Netherlands combined with historical traffic data can be investigated, to see which alternative routes are reasonable in the case of a traffic jam. In the development of KRIS, the second approach was chosen. The reason to do this is as follows. Travelers are not able to monitor the routes, they did not choose. When they have chosen an alternative route, afterwards they do not know whether it was faster then the original route or other alternative routes (unless they know another traveler, who tried the alternative route at the same time). Since the traveler will try different routes on different days he possesses some averaged information, but every time he tries an alternative route he will not know whether it

was the correct one at *that* moment. Besides this, the perception the traveler has of the quality of alternative routes that he tried can be wrong, since it may also be influenced by other incentives then the shortest travel time.

6.6 Conceptual design

The conceptual model is a model of the object system (the system that has to be modeled), which can be created by abstracting and structuring the information obtained during the knowledge elicitation process. The model describes the domain knowledge in a way that is independent of the implementation. Consequently, the conceptual model was the first step in the development of KRIS' expert system. By investigating the maps of the road network and historical travel times the domain knowledge was elicited. Afterwards, this knowledge was made explicit in sets of rules for each freeway. These rules are a model of the knowledge the expert system should possess. They are independent of any implementation thoughts and are the starting point of the model in which implementation issues will be solved, the 'design' model. In this section the conceptual model will be discussed.
As was stated in the previous section, rules have to be made that give alternative routes in the case of congestion along freeway sections. To be able to construct such rules, first has to be clear at what level of 'detail' a road in the network is seen: how long is each section or how many sections contains a road. It has to be decided if rules are constructed for congestion between two ramps, between a certain amount of kilometers or for other sections. Secondly, it should be clear when it is a question of congestion along such a freeway section: which amount of delay is significant enough to try an alternative route.
For the level of detail in which congestion in the road network is monitored route sections between junctions where one can change freeways were chosen. This choice was based on one of the requirements: routes are only optimized in the freeway network (and not considering secondary roads), so only at junctions the route can be changed. Since secondary roads are not taken into account, it is not interesting to construct rules on the basis of congestion between two ramps. For all ramps that are between junctions, the same rules would be constructed, since only at the first junction after each ramp it is possible to change the route. In Figure 6.1 the different road parts between junctions can be found (the junctions are identified by their names).

(entrance coenplein nieuwe_meer)
(route nieuwe_meer badhoevedorp)
(route badhoevedorp burgerveen)
(route burgerveen prins_claus)
(route prins_claus ypenburg)
(exit ypenburg kleinpolderplein)

Figure 6.6: A route from Amsterdam to Delft and the format that is given to the expert system.

It was chosen to keep the dividing line that defines whether there is congestion along a part of the network, outside the domain of the expert system. Traffic data is handled by a separate module of KRIS (see chapter 8), concordant to the modularity requirement. In this module it is decided if the delay is significant enough to consider the part of the road as congested. A discussion on this subject can be found in section 8.6. The expert system is provided with a number of route parts, that each has a label, which can be 'route', 'file', 'entrance' or 'exit'.

Two junctions delimit these route parts. An example of such a route can be found in Figure 6.6. The route on the map was generated by a static route planner and was 'translated' to a route for the expert system. Right of the figure the translation of the route can be found.
Now for each part along the road network rules had to be made that state which alternative route to take if the part was congested. Since there are 92 edges in the network that is monitored by the MONICA system (see chapter 8), the construction of these rules was a time consuming job. The alternative route that can be taken when a route part is congested depends on the direction one is coming from and the direction one is going to. As a result, the rules for alternative roads depend on the previous and following route parts. In Table 6.1 the rules for congestion along the A4 between the 'Prins Clausplein' and 'Badhoevedorp' junction are given. For each route section such a table was constructed. Along the vertical axis the junctions one is coming from are given, while along the horizontal axis the junctions one can be going to are given. When this table is filled all possible alternative routes for different routes containing this trajectory are given. In section 6.8, the implementation of this table into rules of the knowledge base is discussed.

Table 6.1: The rules in case of traffic jam between the Prins Clausplein and Badhoevedorp junctions. GOU denotes the Gouwe junction, KLP the Kleinpolderplein junction, HOL the Holendrecht junction, NM the Nieuwe Meer junction, etc.

from/to	NM (-ams-wat-diem-eem)	NM (ams-wat)	NM (-cp)	NM (-ams-hol)	HOL
GOU	gou-oud-hol-diem-eem; gou-oud-lun-rijns-eem-diem (only if exit eem-diem)	gou-oud-hol-ams-wat	gou-oud-hol-ams-nm-cp; gou-oud-hol-ams-wat-cp; gou-oud-hol-diemwat-cp	Gou-oud-hol-ams	gou-oud-hol
KLP (from gouwe)	(klp-) tb-gou-oud-hol-diem-eem; gou-oud-lun-rijns-eem-diem (only if exit eem-diem)	(klp-) tb - gou-oud-hol-ams-wat	(klp-) tb-gou-oud-hol-ams-nm-cp; (klp-) tb-gou-oud-hol-ams-wat-cp; (klp-) tb-gou-oud-hol-diem-wat-cp (when exit cp-nm then add cp-nm)	(klp-) tb-gou-oud-hol-ams	(klp-) tb-gou-oud-hol
KLP (from r'kerk)	(klp-rid) -gor-ever-lun-rijns-eem-diem; (klp-rid) –gor-ever-oud-hol-diem-eem; (klp-rid) -gor-ever-oud-hol-ams-wat-diem-eem	(klp-rid)-gou-gor-ever-oud-hol-ams-wat-	(klp-rid) –gor-ever-hol-ams-nm-cp; (klp-rid) -gor-ever-hol-ams-wat-cp; (klp-rid) –gor-ever-hol-diem-wat-cp	(klp-rid) – gor-ever-oud-hol-ams	(klp-rid-)-gor-ever-oud-hol

The different steps, which are needed to construct the different rules for each box in the table, are given in the following action list.

For each trajectory:

1. Determine the possible directions where one can be coming from (in the table the junctions along the y-axis),
2. determine the possible directions where one can be going to (in the table the junctions along the x-axis),
3. determine the different alternative routes for each possible route (each box in the table), by investigating the map,
4. calculate the 'detour time' of each alternative: the time needed to make the detour in the best case (no congestion along the alternative route),
5. order the different routes according to this 'detour-time' (incremental requirement),
6. when the 'detour-time' is too large, do not use the alternative,
7. check with reports of car drivers if no routes are missing.

The first two steps are very straightforward, the possible directions can be found by having a look at the map. The third step requires by far the most time: in this step the different

alternative routes have to be chosen. When these routes are known, their travel times can be computed.
Step 5 is important to satisfy the incremental requirement. This requirement states that as fast as possible an alternative route should be available. Searching should continue and if a better alternative route is found this route should be given instead. Consequently, it is important the alternative route that probably will be the best route is computed first. This is the alternative route that has the shortest travel time, unless the probabilities of congestion along the trajectory for which alternative routes are searched and this alternative route are (positive) dependent. This implies that congestion along the trajectory, for which alternative routes are searched, interacts with congestion along this alternative route. This might be the case when, for example, two highways come together at a junction, and continue there as one highway. When there is congestion along the part where the highways have come together, there probably will be congestion along both highways before they come together. When the original route travels along one of these highways and an alternative route for this route is searched because of congestion along this highway, the alternative route along the other highway probably will be congested too. This is illustrated in Figure 6.7.

Figure 6.7: Example of trajectories on which congestion is dependent.

Consequently, the order in which the alternative routes are calculated should depend on the travel times and the probability of congestion along these alternative routes. These probabilities are assessed using historical traffic data. In step 6, it is stated, alternative routes which detour time is too large, should not be investigated. To determine if an alternative route takes too long, there has to be a maximum 'extra' time, above which an alternative route is not incorporated in the expert system. The question is what value this maximum value should be. Several options exist:

- Percentage of the travel time (relative boundary),
- fixed number of minutes (absolute boundary),
- no maximum time is stated, only a maximum number of alternative routes.

The value that is chosen for this boundary will be of critical importance for the actual alternative routes that are found. One the one hand, this value should not be too small, since alternative routes that have quite a long 'detour time' might become the fastest one if all other freeways are congested. On the other hand, this value should not be too big, since otherwise too much alternative routes are generated and have to be calculated: when the value is set to infinity, all alternative routes are calculated, which actually results in another version of Dijkstra's algorithm, only with a poor performance. Consequently, it can be noticed that the value of this boundary has its impact on the quality of the found solutions, especially if severe congestion occurs along some highways, while others are without congestion. As could be expected, a lower value results (on the other hand) in a much better performance. The merits and drawbacks of a low or high boundary value are given in Table 6.2.

Table 6.2: Merits and drawbacks of a low or high boundary value.

	Low boundary value	*High boundary value*
Number of alternative routes	Low	High
Quality of found solution	Good, while no severe congestion along many freeways	Better, although no guarantees can be made
Performance	Good	Less

To overcome the problem of having to choose a boundary value, another approach was chosen. For each alternative route, the 'detour-time' was calculated and incorporated in the rules: the alternative route is only generated, if the delay on the current 'static' route is larger, then the 'detour-time' of the alternative route. In this way the both the qualities of the low and high boundary values are guaranteed.
Of course, still a choice has to be made during the process of constructing the rules of the expert system: which alternative routes are being incorporated in the rules. This choice was made by investigating the map carefully. All alternative routes that show (a little bit) potential are generated and alternative routes that humans would never think of are not incorporated. An example of such an alternative route is driving via Rotterdam, when the highway between Utrecht and Arnhem is congested.

6.7 Design model

In the incremental model the development of the design model is the next step after the conceptual model has been made. The design model still has some degree of abstraction. It is a refinement of the conceptual model, in which implementation issues have been dealt with. When developing the expert system of KRIS, the design model was not made very detailed, since the tables that were made for the conceptual model are already quite detailed. Only one major implementation issue had to be dealt with: the way the reasoning mechanism works. This is discussed extensively in this section. During this discussion two examples are given of how the expert system would work when using both reasoning mechanisms. These examples will also clarify how the tables that were mentioned in the previous subsection will be implemented.
There are two possibilities for the reasoning mechanism used by the expert system to generate alternative routes. The first one is to let the expert system do several iterations to find alternative routes. First the original static planned route is given to the expert system. For each trajectory between junctions that have severe congestion one or more alternative routes (if possible) are generated. This implicates, that the rule base of the expert system contains rules, that give the best alternative route for each congested trajectory, given the next and previous junctions (the tables that were discussed in the previous paragraph). When more then one section of the route is congested, the different alternative routes that are found for the different congested sections are combined, if possible, to routes and are used 'alone' to construct different alternative routes. Now these routes are investigated, to see if there are any congested trajectories in these routes. The routes that (still) contain congested trajectories are offered one by one to the expert system again, and the above described process takes place again. These steps are repeated until all possible alternative routes have been generated and the best alternative route is found. This algorithm is illustrated in the first example, where also the merits and drawbacks of this method are shown.
Another possibility for the operation of the expert system is to generate all possible alternative routes at once (not only the best alternative route, as in the above algorithm), taken into account the possibility that parts of the generated alternative route also can be congested. In this case the rule base of the expert system contains far more elaborate rules, since all alternative routes have to be generated for each congested trajectory and not only one alternative route. On the other hand, only one iterations of the expert system is needed, since all possible alternative routes are generated in one time and the drawbacks of the first mentioned method do not exist. This approach, and its merits and drawbacks, is illustrated in the second example. This section concludes with an overview of the merits and drawbacks of both implementations, which can be found after both the examples.

Multiple iterations example
A journey request from Bleiswijk to Zaanstad is done. The static route planner finds a route that begins at the ramp Bleiswijk at the A12 in the direction of The Hague, switches to the A4

at the Prins Claus junction, follows the A4 until the Nieuwe Meer junction, then continues along the A10 to the Coenplein junction, from where Zaanstad is reached. This route is illustrated in Figure 6.8. Unfortunately the trip has to be made during a very intense rush hour: along all parts of the route are severe delays, and all trajectories between junctions get the label 'file'. This means the expert system gets the following route at its blackboard: (the trip after the Coenplein is not mentioned, since the expert system is not working for this part of the road network yet, see also section 6.4 and Figure 6.1).
For every congested trajectory one of the rules in the rule base will be instantiated. Every rule fires the best alternative route, as was stated in the beginning of this section. In Table 6.3 the results of this firing process of each rule are given.

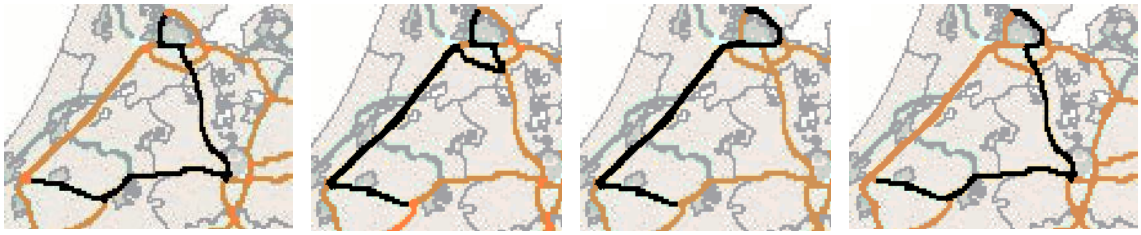
(part entrance gouwe prins_claus)
(part file prins_claus badhoevedorp)
(part file badhoevedorp nieuwe_meer)
(part file nieuwe_meer coenplein)

Figure 6.8: Route from Bleiswijk to Zaandam and the expert system translation.

Table 6.3: Alternative routes found for different congested trajectories.

file Prins Claus Badhoevedorp	file Badhoevedorp Nieuwe Meer	file Nieuwe Meer Coenplein
Prins Claus	(Prins Claus)	(Prins Claus)
Gouwe	Badhoevedorp	(Badhoevedorp)
Oudenrijn	Holendrecht	Nieuwe meer
Holendrecht	Amstel	Amstel
Amstel	Nieuwe meer	Watergraafsmeer
Nieuwe meer	(Coenplein)	Coenplein

The three alternative routes are illustrated in Figures 6.9 to 6.11. Immediately one of the drawbacks of this approach becomes clear: it is very difficult to combine the alternative routes that are given into one route. The only possible combination of the alternative routes is the first alternative route, combined with the third one. First the route will follow the A12 and A2 to the Amstel junction, and then it will continue along Watergraafsmeer to the Coenplein junction. The route in the Figure 6.12 illustrates this route. A human being can see this combination quite quickly, but for a computer program it will be very hard to combine different alternative routes in a way that makes sense.



Figures 6.9 to 6.12: The first three figures illustrate the different alternative routes of Table 6.3. The last figure illustrates the combined route.

When there is no congestion along the alternative route suggested, the reasoning process can be stopped now. But when there is also congestion along the alternative route, the route search algorithm has to be applied (again) to all these routes. For example, when there is congestion between Holendrecht and Amstel, between Amstel and Watergraafsmeer and between Watergraafsmeer and the Coenplein junction, all found routes have to be examined again: the expert system has to be ran again for each of the alternative routes. Since it would take pages of paper to show this process for each of the routes, it is only shown for the first alternative route. First the route that is given to the expert system is shown and in the table beneath it (Table 6.4) the different alternative routes, the expert system should give, for the three congested trajectories are given.

(entrance prins_claus gouwe)
(route gouwe oudenrijn)
(route oudenrijn holendrecht)
(file holendrecht amstel)
(file amstel watergraafsmeer)
(file watergraafsmeer coenplein)

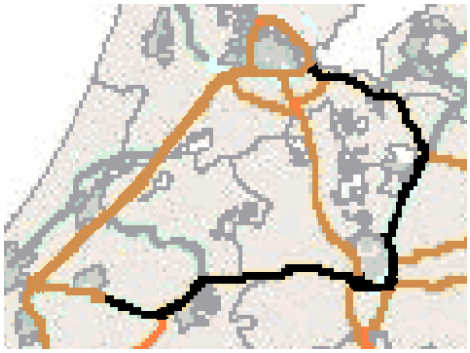


Figure 6.13: Alternative route along the A27.

Table 6.4: Results of the second iteration of the expert system.

file Holendrecht Amstel	file Amstel Watergraafsmeer	file Watergraafsmeer Coenplein
Holendrecht	Amstel	Watergraafsmeer
Badhoevedorp	Holendrecht	Amstel
Nieuwe_meer	Diemen	Nieuwe_meer
Amstel	Watergraafsmeer	Coenplein

Now again the same procedure as before has to be applied. First has to be tried to combine the different alternative routes. As one can see this is again a difficult process. Probably the best combination that should be tried apart of the three found alternative routes is the second route, but starting from the Holendrecht junction. While a lot of junction names are the same in the different routes, other combinations do not make sense: this is the hardest part of trying to make combinations of different routes.
One can see that the alternative routes found still contain trajectories that are congested. This illustrates another drawback of the method: it is not clear when the process of trying to find new alternative routes can be stopped. A possibility would be to consider the trajectories that already have been considered by previous iterations of the expert system not any more. Unfortunately, this approach also has a major drawback, since alternative routes that are only useful in the case of congestion along different parts of the road are not considered any more. An example is illustrated in Figure 6.13, where the alternative route along the A27 in the case of a major congestion, is showed.
The last, but most important drawback of this method, which is also illustrated very well in this example, is that the expert system is instantiated very often. While there were only two ‘iterations’ the expert system was ran five times. If there had been another iteration, this number would have grown exponentially. Every time the expert system is instantiated and run, this takes a considerable amount of time, which slows down the computation time considerably.

One iteration example

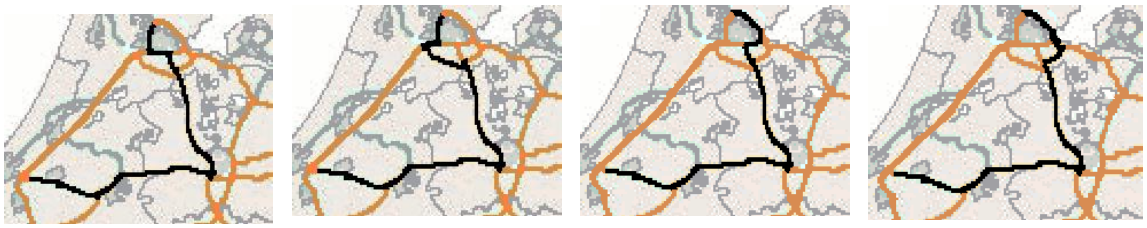
The same route as was used in the multiple iterations example is used. Now for every trajectory of the route that is congested *all* possible alternative routes are generated. In Table

6.5 the firing results of the rule that was instantiated by the congestion along the trajectory between the Prins Clausplein and Badhoevedorp are illustrated.

Table 6.5: Results of the rule that is instantiated by (file prins_claus badhoevedorp)

Route 1	Route 2	Route 3	Route 4
Prins Claus	Prins Claus	Prins Claus	Prins Claus
Gouwe	Gouwe	Gouwe	Gouwe
Oudenrijn	Oudenrijn	Oudenrijn	Oudenrijn
Holendrecht	Holendrecht	Holendrecht	Holendrecht
Amstel	Badhoevedorp	Amstel	Diemen
Nieuwe meer	Nieuwe meer	Watergraafsmeer	Watergraafsmeer
Coenplein	Coenplein	Coenplein	Coenplein

One can see that the alternative routes that are given are the same as were given after one or more iterations in the multiple iterations approach. In Figures 6.13 to 6.17 these different alternative routes are shown. As one can see, the fastest alternative route would be route 1. The other alternative routes are given too, because of the possibility of a traffic jam on the road between the Holendrecht and Coenplein junction.



Figures 6.13 to 6.17: Different alternative routes in case of congestion along the Prins Claus – Badhoevedorp trajectory.

The three alternative routes for a traffic jam between the Badhoevedorp and Nieuwe Meer junctions and between the Nieuwe Meer and Coenplein junctions are given in Tables 6.6 and 6.7. The junctions that are between ( ) are not given in the alternative route, but are added by the Java code.

Table 6.6: Results of file Badhoevedorp Nieuwe Meer.

Alternative route 5	Alternative route 6	Alternative route 7
(Prins Claus)	(Prins Claus)	(Prins Claus)
Badhoevedorp	Badhoevedorp	Badhoevedorp
Holendrecht	Holendrecht	Holendrecht
Amstel	Amstel	Diemen
Watergraafsmeer	Nieuwe meer	Watergraafsmeer
Coenplein	(Coenplein)	Coenplein

Table 6.7: Results of file Nieuwe Meer Coenplein.

Alternative route 8	Alternative route 9	Alternative route 10
(Prins Claus)	(Prins Claus)	(Prins Claus)
Badhoevedorp	Badhoevedorp	Badhoevedorp
Nieuwe_meer	Holendrecht	Holendrecht
Amstel	Amstel	Diemen
Watergraafsmeer	Watergraafsmeer	Watergraafsmeer
Coenplein	Coenplein	Coenplein

As can be seen, only one expert system phase is needed to find all alternative routes. This is a very useful property, since it reduces the computation time significantly and an upper limit

of the computation time of the expert system can be given. The drawbacks of this method can be noticed too: a lot of exhaustive search and type work has to be done, to be able to generate all possible alternative routes for a given trajectory that is congested: the best example is in the first four alternative routes. Actually only one 'real' alternative route is given for the congestion along the A4 between the Prins Claus and Badhoevedorp junction. The other alternative routes are only useful in case there is congestion between Holendrecht, Amstel and the Coenplein junction. If there was no congestion along this route, the other three alternative routes are not interesting any more.
Another drawback of this approach, that several routes can be found twice or even more times, is also illustrated in this example: two of the ten alternative routes found are identical and are useless because of this (alternative 5 = alternative 9 and alternative 7 = alternative 10). It can be concluded that the number of found alternative routes that is actually useless depends mainly on the number of congested trajectories along the road network. Only two routes were found that were identically, while 7 found routes would have been useless if the three files between the Prins Claus, Badhoevedorp, Nieuwe Meer and Coenplein junctions would have been the only ones in the part of the road network that is used by the alternative routes. As will be stated later in this section, when the traffic reports of the Dutch road network are studied, it can be shown, that during the rush hours (when almost all files of a day are reported) a lot of freeways are congested. As a consequence, most alternative routes that are generated by this approach are useful after all.
As could be expected, for KRIS the one iteration approach (the last one described) was chosen, because of implementation considerations. When a considerable part of the trajectories of the road network is congested, which is often the case in the rush hours in the Netherlands, this means the multiple iterations approach (first one described) has to apply several expert system iterations to find the optimal route. The performance of the system will degrade significantly using this approach, since every time the expert system has to be invoked, run and read out again. Every iteration all rules have to be compared with the facts that are put onto the blackboard, while in the latter approach these comparisons are made only once, for the same amount of rules. Secondly, it is not clear when the iteration process can stop (e.g. when all alternative routes have been generated), while the number of invocations of the expert system grows exponential with the number of iterations. On the one hand the number of iterations has to be large enough to generate all alternative routes, while on the other hand the number of iterations has to be kept as low as possible to keep a good performance of the system. In the multiple iterations example these problems are illustrated very clear. Finally it is very difficult to combine different alternative routes, as mentioned in the first paragraph, since the different alternative routes for different congested trajectories often cannot simply be appended, because of the fact the finish and start junctions are not the same. Very sophisticated intelligence is needed to perform this process. This drawback is also illustrated quite well in the example.
The main drawbacks of the chosen - one iteration - approach are twofold. Firstly, it is possible that a lot of alternative routes are generated that are useless: when the 'best' alternative route is free of congestion, all the other routes that are generated are tested, while they are not used. So in the case of little congested roads (for example only the one or more trajectories of the static route) the algorithm is not very efficient. But in the Netherlands this is (unfortunately) not often the case. When studying statistics about congestion on the Dutch road networks it can be concluded that congestion is at a large percentage of the roads during the rush hours, while there is almost no congestion during the other hours of the day [vid01]. Secondly, the construction of the rules in the rule base is a lot more complex then when using the first approach. For each trajectory with congestion all alternative routes have to be generated, taking into account all possible congested trajectories that can be used. Of course this is not a very severe problem, since this is only a 'one time problem'. Finally, another drawback of this approach is that several routes can be found twice or even more times, if there is a traffic jam at more trajectories, since the rules for each congested trajectory also

give alternative routes in case of congestion at trajectories of the route it constructed. If the constructed route contains parts that were also in the original route, it is possible different rules will construct the same alternative routes for this trajectory.
For clarity the pros and cons of each approach are summarized here:

Multiple iterations approach:

- *Merits:*
  - If there is no other congestion then on the trajectories of the static routes, no useless alternative routes are tried,
  - simple, clear rule base.
- *Drawbacks:*
  - Performance degrades when more then one iteration is needed,
  - in the case of many trajectories are congested it is not clear when to stop the iterations,
  - increasing number of iterations implicates exponential growth of expert system invocations,
  - sophisticated intelligence is needed to combine different alternative routes.

One iterations approach:

- *Merits:*
  - Only one iteration needed: very efficient if there is a lot of congestion and it is clear when to stop the reasoning process,
  - no combination of different alternative routes needed, so no elaborate coding needed for this process,
  - no possible exponential growth of the time needed to construct an alternative route in the case of congestion along a large part of the road network.
- *Drawbacks:*
  - In the case of little congestion alternative routes are tried that are useless,
  - construction of rules in the rule base is quite complicated,
  - routes can be found twice or even more times.

6.8 Implementation

As was stated in the previous section, the implementation of the tables into rules of the knowledge base is quite straightforward. The only rules that can become quite complicated, are the rules in the tables that depend on more junctions than only the last junction before the current road part and the first one after.
The expert system was implemented using CLIPS (C Language Integrated Production System) and Jess (Java Expert System Shell). CLIPS is a widely known expert system shell [clips]. Since KRIS has been programmed in Java (see chapter 9) the choice of Jess was not so complicated: it is the only know expert system shell for Java. Jess makes it possible to simply load files containing CLIPS rules, to read in a number of facts and then to run the expert system. The results of the expert system run can be read out again very simply using some Java code. Since CLIPS rules can simply be loaded in Jess all rules were first tested in a CLIPS environment. Later the co-operation with the Java code was tested.
In Figure 6.18 an example of the implementation of a rule into CLIPS code is shown. This is the implementation of the first box of Table 6.1, containing the alternative routes in case of congestion between the Prins Clausplein and Badhoevedorp junctions, while coming from the Gouwe junction and continuing along the Amstel, Watergraafsmeer, Diemen and Eemnes junctions. Since this is quite a simple rule and for each box of each table from each trajectory such an implementation has to be made, one can imagine that the process of rule constructing requires a significant amount of time.

As one can see, no travel times are calculated in the rules. This is done outside the expert system. The expert system gets a route with labels and generates different alternative routes. The travel time of each alternative route is computed and compared with the best time found so far. When a better travel time is found, this solution becomes the best one so far. In this way the incremental searching and the modularity requirements are met. The advantage of this modular approach is, that the expert system functions independent of the chosen algorithm to compute (or estimate) the travel times. In chapter 8 travel time estimation is discussed in detail.

```
(defrule filePrinsClausBadhoevedorp1_1 ""
(declare (salience 90))

?r <- (part file prins_claus badhoevedorp ?time)
?r0 <- (part ?type0 gouwe prins_claus ?time0)
?r1 <- (part ?type1 badhoevedorp nieuwe_meer ?time1)
?r2 <- (part ?type2 nieuwe_meer amstel ?time2)
?r3 <- (part ?type3 amstel watergraafsmeer ?time3)
?r4 <- (part ?type4 watergraafsmeer diemen ?time4)
?r5 <- (part ?type5 diemen eemnes ?time5)
=>
(assert (alt ?*altnr* 1 prins_claus gouwe))
(assert (alt ?*altnr* 2 gouwe oudenrijn))
(assert (alt ?*altnr* 3 oudenrijn holendrecht))
(assert (alt ?*altnr* 4 holendrecht diemen))
(assert (alt ?*altnr* 5 diemen eemnes))
(bind ?*altnr* (+ ?*altnr* 1))
(if (= (str-compare ?type5 "exit") 0) then
      (assert (alt ?*altnr* 1 prins_claus gouwe))
      (assert (alt ?*altnr* 2 gouwe oudenrijn))
      (assert (alt ?*altnr* 3 oudenrijn lunetten))
      (assert (alt ?*altnr* 4 lunetten rijnsweerd))
      (assert (alt ?*altnr* 5 rijnsweerd eemnes))
      (assert (alt ?*altnr* 6 eemnes diemen))
      (bind ?*altnr* (+ ?*altnr* 1))
)
)
```

Figure 6.18: Example of CLIPS code for one of the boxes of Table 6.1.

Chapter 7: Graph algorithm

In this chapter the algorithm that was constructed to find the shortest route in a (dynamic) graph is discussed. After an introduction in section 7.1, the algorithm is presented in section 7.2. In this section both a textual and a pseudo-code description are given. In section 7.3 the algorithm is illustrated with an example, while in section 7.4 the strong and weak points of the algorithm will be discussed. The chapter ends with a section about the implementation of the algorithm (section 7.5).

7.1 Introduction.

The most secure way to find an optimal route from a departure to a destination address at a specific time of the day would be to model the problem in a graph and to apply the shortest path algorithm. This would be a very natural approach, since a highway network is a graph. All route planners that do not use dynamic information use a shortest path algorithm to find the optimal route: the Dutch road network is represented in a graph and the shortest path algorithm of Dijkstra is applied. But, as was already stated in section 5.3, in the case of finding an optimal route while using dynamic information problems arise, since the dynamic information has to be modelled. In a 'regular' graph the vertices simply represent the kilometres or travel time needed to travel from one node to another, but this is no longer possible when using dynamic data. It is possible to adjust the travel time to the current road situation (the travel time will be much longer in case of a traffic jam), but it is *im*possible to apply the same algorithm. The road situation changes in time, so the travel times also change. Once a traveller has reached the next node in the graph, travel times may have changed significantly and it may well be possible the chosen route is not the best anymore. This is illustrated in Figure 7.1. Consequently, in some way the dynamic information has to be taken into account by the algorithm. To take this information into account, an algorithm was constructed.

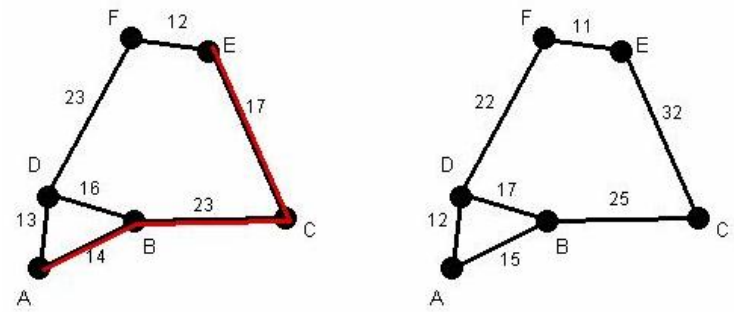


Figure 7.1: Two road networks. At the departure time the fastest route is A-B-C-E (left graph), but 15 minutes later the travel times have changed (right graph) and A-B-C-E is not the fastest route anymore. A-D-F-E would have been much faster and even B-D-F-E is faster now then B-C-E.

7.2 The extended Dijkstra algorithm

To find the best route in a dynamic graph an algorithm was constructed. This algorithm is presented in this section. Since the graph is not a static graph, it was thought Dijkstra's algorithm could not be used and another algorithm was constructed. When the constructed algorithm was reviewed thoroughly, it was discovered it differs with Dijkstra's algorithm

only slightly. Consequently, it was called the 'extended Dijkstra algorithm'. This section starts with a textual description of the algorithm.
The starting point of the algorithm is the directed graph of the road network that is also used for a 'normal' shortest path algorithm. The crucial part of the algorithm is the calculation of the estimated travel time for each edge. First, for the edges starting at the departure point the travel times at the time the traveller wishes to start his journey are calculated. If a traveller stated an arrival time, the algorithm is started from the destination vertex. Now the travel times from the next nodes can be computed, since the arrival time at these nodes is known (the departure time plus the travel time for the edge). These steps can be repeated until the destination node has been reached. Meanwhile two other actions have to be taken. First, cycles have to be removed, if they have a longer travel time to a certain node, than another connection from the departure node to this node. The edges and nodes that construct this cycle (and are not part of another route) have to be removed. Secondly, edges that are part of a route with a travel time longer then another route from the departure node towards the destination (and are not part of another route) have to be removed. Finally, nodes and their outgoing edges that have no incoming edges have to be removed. When these steps are applied in the correct order, the best possible route from the departure node towards the destination node remains. In Figure 7.2 a pseudo code description of the algorithm is given

Given a graph of the road network, a departure time and a starting point A and a destination point B:

1. Compute all estimated travel times from node A to adjacent nodes and compute the arrival times of al these nodes.
2. Starting with the node that has the earliest arrival time:
   For all edges of this node:
   a. Compute the estimated travel time,
   b. if the destination node already has a connection:
      if the path from node A to this node along this connection is shorter: delete this edge
      else delete the other edge,
   c. if a node with no outgoing edges remains that is not the destination node and not in the best path found so far:
      delete the node and its incoming edges,
   d. if a node with no in going edges remains and its not the origin node:
      delete the node and its outgoing edges.
3. While there are nodes remaining that do not have an arrival time: repeat step 2.
4. While there are edges remaining that do not have a travel time:
   For each edge:
   a. Compute the arrival time in the node where the edge finishes,
   b. if the arrival time is earlier then the arrival time the node had before:
      delete the complete path that resulted in this arrival time,
   c. if the arrival time is later then the arrival time the node had before:
      delete the path towards the node where the edge started, unless (parts) of this path contribute to the best solution so far.

Figure 7.2: Pseudo-code description of the algorithm.

As was stated in the introduction of this section, this algorithm only slightly differs from Dijkstra's algorithm. In stead of a travel time to each node the arrival time in the node is stored and the travel times in the above described graph are dependent of the point in time one wants to travel an edge. In Dijkstra's algorithm these travel times are independent and constant. This change leads to one restriction to be able to apply the 'extended' Dijkstra algorithm: the estimated travel time should never decrease faster in time, then the time itself passes by. For example: if one want to travel from node A to B and at 12:00 a.m. and the

travel time is 14 minutes, then it is not possible that at 12:05 a.m. the travel time is less than 9 minutes, since otherwise the traveller that leaves at 12:05 a.m. will overtake the traveller that leaves at 12:00 a.m.. This seems quite a reasonable restriction, since the dynamics of a system like a road network make it impossible to start a trajectory later and finish earlier, without passing the vehicle that left earlier (when both drive the maximum allowed speed). The restriction implicates the estimated travel time data should be tested, since it may well be possible that these data does not confirm to this restriction. As stated in section 5.3 the data are estimated and it is possible these estimates are not as strict as the situation at a road would be.

7.3 Example

In this section an example of the algorithm is given. In the example a reference is made to the numbers of the steps of the pseudo code that was given in the previous section. The graph, which will be used in this example, is illustrated in Figure 7.3. The departure node is node A, while the starting time is 12:00 am. The shortest route to node F has to be found.

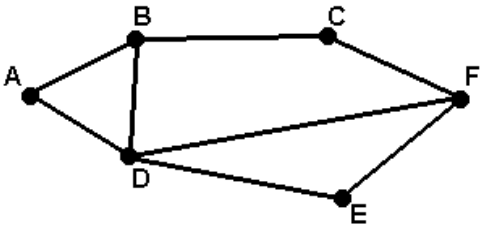


Figure 7.3: The starting graph.

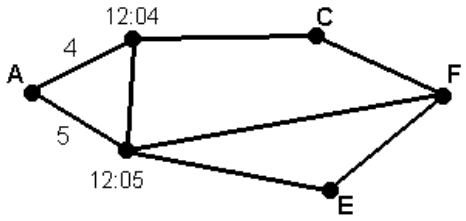


Figure 7.4: The graph after step 1.

First step 1 is carried out. All travel times from node A to the adjacent nodes at 12.00 have to be calculated: A-B is 4 minutes and A-C is 5 minutes, so the arrival times are 12:04 for node B and 12.05 for node C.
As can be seen in Figure 7.4, node B has the earliest arrival time. The next step is to carry out steps 2 a to d for this node. The result of step 2 is illustrated in Figure 7.5.

2a. The travel times are 6 minutes for B-C and 4 for B-D, so the arrival times are 12.10 for node C and 12.08 for node D.
2b. Node D already has a connection. This path is shorter so the edge B-D is deleted.
2c. No nodes with no outgoing edges remain.
2d. No nodes with no in going edges remain.

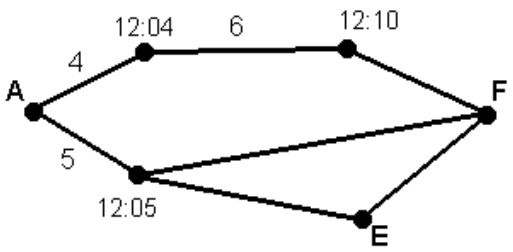


Figure 7.5: The graph when steps 2a to d. are carried out for node B.

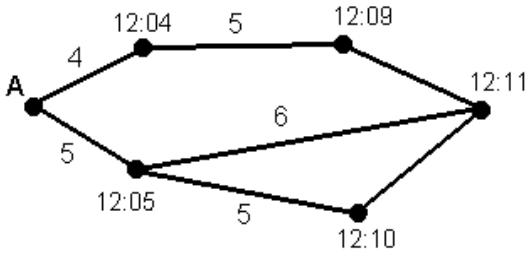


Figure 7.6: The graph when steps 2a to d are carried out for node D.

According to step 3 of the algorithm, for all nodes that do not have an arrival time yet, steps 2a to d have to be carried out, starting with the node with the earliest arrival time, which is node D. The results of steps 2a to 2d are illustrated in Figure 7.6.

2a. There are two outgoing edges in node D: D-E and D-F. Their travel times are resp. 5 and 6 minutes, so the arrival time will become 12.10 for node E and 12.11 for node F.
2b. Node E does not have any incoming paths yet, so no edges or nodes are removed, neither does node F.
2c. No nodes with no outgoing edges remain.
2d. No nodes with no in going edges remain.

Now all nodes have an arrival time, but the edges C-F and E-F do not have a travel time yet. For both edges, steps 4a to 4c have to be carried out. First this is illustrated for edge C-F (see also Figure 7.7), then for edge E-F (see also Figure 7.8).

4a. The arrival time of node F (where C-F) finishes, will become 12:13, since the travel time between node C and F at 12:09 is 4 minutes.
4b. This arrival time is later then 12:11, so the original path will not be removed.
4c. Since the arrival time is later, the path to node C will be deleted. This is illustrated in Figure 7.5, which shows the graph after these steps.

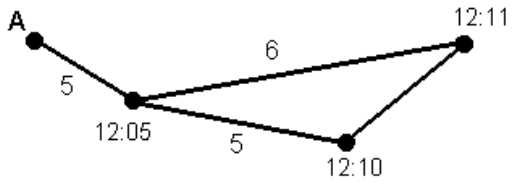


Figure 7.7: The graph after steps 4a to 4c for edge C-F.

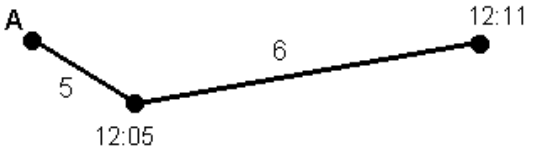


Figure 7.8: The graph after steps 4a to 4c for edge E-F.

In Figure 7.7 can be seen that only edge E-F has no travel time yet. So steps 4a to 4c will be carried out again for this edge

4a. The arrival time of node F, travelling via node E, will become 12:13, since the travel time between node E and F, when departing at 12:10 is 4 minutes.
4b. This arrival time is later then 12:11, so the original path will not be removed.
4c. Since the arrival time is later, the path to node E will be deleted, except for the part that belongs to the best found solution so far: edge A-D. This is illustrated in Figure 7.8, which shows the graph after this step.

As can be seen in Figure 7.8, no edges without a travel time or nodes without an arrival time are left, so the algorithm is finished. The best route from node A to F when departing at 12:00 is via node D, and takes 11 minutes.

7.4 Properties of the extended algorithm

When the algorithm is reviewed in the light of the problem definition, which was given in section 5.2, it seems that the 3 dimensional matrix (section 5.2.1) is not used. Actually, this is not the case. It can be noticed, the 3 dimensional matrix is not constructed before the algorithm starts searching, but during the application of the algorithm the parts of the 3 dimensional matrix that are needed are calculated. Every time the travel time of an edge is needed, the travel time is estimated.
As was already stated in section 5.4, the 3 dimensional matrix is not calculated, since this calculation is a very time consuming process. Every minute all travel times between the nodes would have to be calculated, while only some of these calculations are actually used

by the searching algorithm. Consequently, these travel times are only calculated if the extended Dijkstra algorithm needs them. In Figure 5.7 this was illustrated.
It can be concluded, the extended Dijkstra algorithm does search the shortest path in the 3 dimensional matrix, but without constructing the complete matrix. This leads to the most important property of a solution that is found by this algorithm: the fact a found path is guaranteed the shortest route, given correct estimates of the travel times.
Unfortunately, this property also implicates some drawbacks. To be sure the best solution is found almost all paths from the starting node to the destination node are tried. Paths are only rejected at the moment the travel time of the sub-path found so far becomes larger then the best path found so far. Consequently, the travel time of each edge has to be calculated often, since different paths to the starting node of each edge will lead to different departure times for each edge. As can be read in chapter 8, where the estimation of travel times is discussed, the estimation of travel times is a very time consuming process (which is also the reason no 3 dimensional matrix is created). Consequently, to have a good performance the number of travel time estimates should be as low as possible. This implicates a serious drawback of the algorithm, since it requires a lot of travel time estimates.
Apart from the time needed to calculate the travel times the following observation can be made about the algorithm. The running time of Dijkstra's algorithm can be expressed as a function of the vertices (n) and a good implementation of the algorithm will take $O(n^2)$ time [good98]. This means the computation time increases quadratic with the number of vertices. When the graph that is being considered becomes large, this implicates the computational time may increase significantly, but probably this will not influence the over all computational time too much, since the estimation of the travel times requires the most time.
A summary of the merits and drawbacks of the extended Dijkstra algorithm can be found in Table 7.1.
It can be concluded that the guarantee of the best solution has to be paid for. Apart from the quadratic increase of computation time in the number of vertices, the calculation of the estimated travel times might slow down the algorithm significantly. In the prototype of KRIS, the number of travel time estimates will be counted and the computation time will be measured, to be able to make a comparison with the expert system approach. The results of these comparison will be discussed in chapter 10.

Table 7.1: Merits and drawbacks of the extended Dijkstra algorithm

Merits	Drawbacks
Guaranteed best solution	Very often a computation of the travel time is needed.
	$O(n^2)$ computation time in vertices

7.5 Implementation

According to the modular design and the OO model (see Appendix A) a separate class has been used to implement the extended Dijkstra algorithm. Since the extended Dijkstra algorithm only differs slightly from the original algorithm, a package was used that also contained an implementation, the Java Data Structure Language (JDSL) package [good99]. Due to the object-oriented properties of this class a significant number of classes could be used without changes (for example the Vertex and Node classes), while some changes were made to the algorithmic classes. It can be stated that the JDSL package simplified the implementation task significantly and reduced the development time. Nevertheless a lot of time was needed to make the essential changes and to test the algorithm.

Chapter 8: Estimating travel times

It will be clear that the performance of KRIS will largely depend on the quality of the estimated travel times. Alternative routes are compared on the total estimated travel times, thus if these travel times are not estimated correctly the wrong alternative route might be chosen. In this chapter possible ways to estimate and predict travel times are discussed. Firstly, in section 8.1 the monitoring system MONICA that is available in the Netherlands is introduced, which might be used to provide KRIS with real time data. In section 8.2 a clear distinction between travel time estimation and prediction is made. In the following sections several techniques to provide KRIS with the needed travel times are introduced: using estimated travel times (section 8.3), predicted travel times (section 8.4) and using historical data (section 8.5). Since there were no MONICA data available during the development of KRIS, other data were used to provide KRIS the travel times. In section 8.6 a description of how these data were collected is given.

8.1 Monica

During the last two decades a monitoring system was placed along all major Dutch highways to be able to detect incidents. This system was called MONICA, after MONitoring CAsco system and was installed by the Department of Public Works. Major reasons to implement this system were [reijm99]:

- Better use of available capacity by:
  - a better distribution of traffic over all highways,
  - preventing local overload, which can result in significant loss of capacity.
- To improve safety by:
  - decreasing the chance at primary incidents,
  - warning in the case of traffic hold-up, which will decrease the chance at secondary incidents.
- To lighten the task of the road authorities and police by:
  - offering tools to carry out roadwork more efficient,
  - offering possibilities to take measures quickly in the case of accidents.
- To collect traffic data, to:
  - make an evaluation of the system possible,
  - optimize the operation of the system using the measured traffic data,
  - develop new control strategies that can be added to the system.

In Figure 8.1 the MONICA system is illustrated. Major components of the system are the detection stations. Every 500 meters two induction loops have been placed in each lane. The data of these loops are collected by the detection stations, which each have their own microprocessor that processes the data and sends these data to the substations. The substations are placed beneath the signal signs they control. Each substation processes the data of maximal three detection stations to perform automatic incident detection (AID) and communicates with a central processor to send data when this processor asks for them. When communication falls out the substation runs its own program to perform automatic incident detection and to show the right signs. The central processor takes care of the collection of raw detector data, stores these data and sends the data to the TIC's.
At the so called Traffic Information Centers (TIC) the speed, intensity and flow measurements that have been collected by the central processor are translated into queue reports. These queue reports are distributed using radio, Internet, teletext, etc.. Since a few months these queue reports are also used on so-called Dynamic Route Information Panels

(DRIP's), which provide the driver with queue–length information of route alternatives ahead.

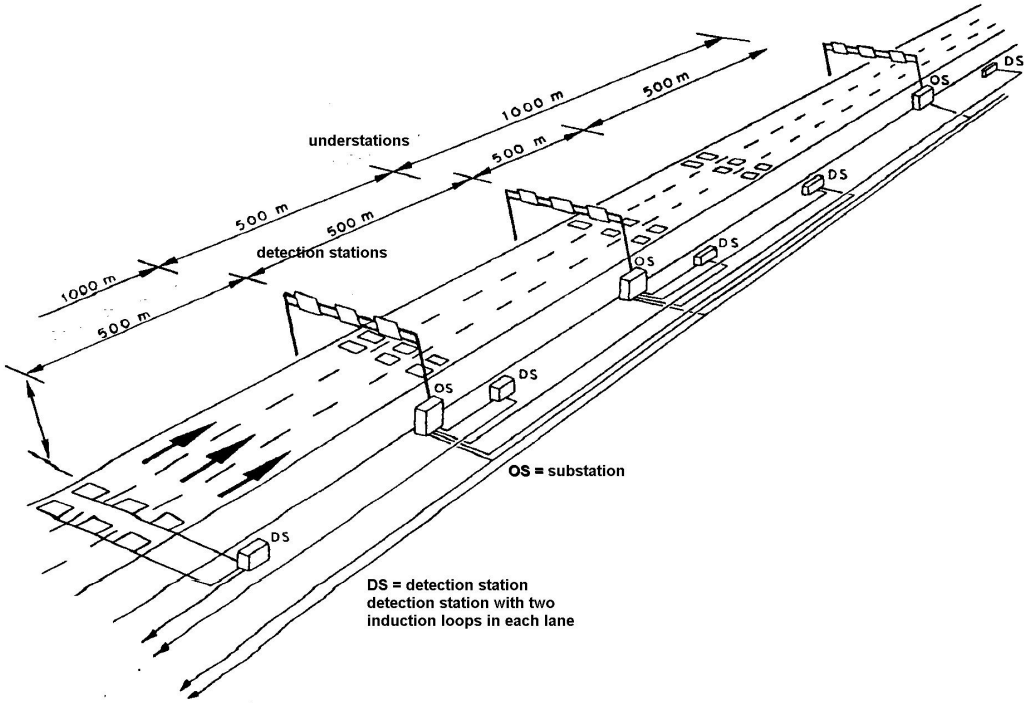


Figure 8.1: System configuration of MONICA.

The data MONICA generates can also be used to estimate travel times. These estimated travel times could be used in KRIS' algorithm to find the best route at a certain moment. At the moment the Department of Public Works is developing a system, called MONIBAS (MONIca BASisapplicaties), which will calculate and provide these estimated travel times based on the data MONICA provides. One could imagine that when KRIS is coupled to MONIBAS a perfect route planner could be constructed. Unfortunately, this is not the case yet, since the travel times this system provides are not accurate enough yet. As will be explained in the next section these travel times are estimations, while travel time predictions are needed. Another possibility, which was already discussed in section 3.8, is that the travel time predictions are inferred from the positions of the KRIS users. Then the MONICA or MONIBAS data are not needed anymore, since KRIS can function completely independent.

8.2 Travel time estimation and prediction

A distinction is made between travel time estimation and travel time prediction. In this section a definition of both terms is given according to [thijs00].
Estimation refers to the calculation of traffic state variables, for the most recent period for which measurements of the traffic are available. This implies that the experienced travel times on a section are determined after the vehicles have passed the section. The section level travel time is denoted as SLTT. If the current period is denoted with p, estimated traffic state variables such as travel time may be established for period p' (p'<p) at best, based on traffic measurements from period p' or earlier:

$$SLTT_k(p') = f(X(p', p'-1, p'-2, \ldots)) \qquad p' < p$$

for the k-th section, where X is a vector of measured traffic variables e.g. speeds.

In section 8.3 an introduction to techniques to estimate the current travel time is given.

Prediction refers to the calculation of future traffic state variables. Travel time predictors calculate travel time values for current and future time periods by using predicted traffic state data. This definition implies that the experienced travel times on a section are determined before the vehicles have entered this section. These future traffic state data are predicted using prediction models, which are calibrated to the traffic system under consideration.

$$SLTT_k(\hat{p}) = f(Y(\hat{p}, \hat{p}-1, \hat{p}-2, \hat{p}-3...)) \qquad \hat{p} \geq p$$

where Y is a vector of model-based predicted traffic variables, such as flows, densities, etc.

In section 8.4 an introduction to travel time prediction techniques will be given.

8.3 Travel time estimation

At the moment all traffic information distributed in the Netherlands is based on estimated travel times. Queue reports all report the *current* situation at the highway network. As was stated in section 8.1 a new system is being installed, called MONIBAS, which estimates (current) travel times. In this section the way travel times are estimated is explained, since it illustrates the complex calculations needed to estimate travel times.
Two possible methods exist to estimate travel time: using instantaneous section level travel time estimators (SLTT's) and using dynamic SLTT's. The latter approach can only be applied off-line, since traffic data for a number of periods is needed. Consequently, in this section travel time estimation using instantaneous SLTT's is discussed: KRIS requires on-line travel estimation. The remainder of this section is based on [thijs00].
The objective of travel time estimation is to approximate as closely as possible the travel time a traveller experiences on his route when starting at instant t. The experienced travel time is defined by the following expression:

$$\hat{T}_R(t) = \int_0^L \frac{1}{v(x,\tau)} \qquad \tau > t, \text{ where } v(x,\tau) \text{ is the local speeds at } (x,\tau) \text{ and L is the route length of route R.} \qquad (8.1)$$

An instantaneous travel time estimator takes only account of the local speeds in the sections of a route that prevail at the instant of departure t. On the contrary a dynamic travel time estimator takes account of the locally prevailing speeds during travelling the section of a route. Consequently, the instantaneous travel time is defined by the following expression:

$$T_R(t) = \int_0^L \frac{1}{v(x,t)} dx \qquad \text{where } v(x,t) \text{ is the local speeds at } (x,t) \qquad (8.2)$$

Two methods to estimate the instantaneous SLTT will be discussed:

- speed-based SLTT estimator,
- SLTT estimator based on a mass-balance.

The speed-based SLTT estimator makes use of the assumption that vehicles drive half of the section with the time-mean speed detected at the downstream end of the section, $V_{a,k}(p)$, and half of the section with the time-mean speed at the upstream end of the section, $V_{b,k}(p)$, (at both ends detection loops are placed that measure the time-mean speed, see section 8.1) which is expressed in equation 8.3 and illustrated in Figure 8.2.

$$SLTT_k^{inst}(p) = \frac{L_k}{2V_{a,k}(p)} + \frac{L_k}{2V_{b,k}(p)} \qquad \text{where } L_k \text{ is the length of section k} \qquad (8.3)$$

The SLTT estimator based on mass-balance compares the inflow of a section during a previous time period, $I_{a,k}(p\text{-}t_k^{freeflow})$, with its outflow during the current time period: $I_{b,k}(p)$. The time offset is ideally set to the average free-flow travel time $t_k^{free\text{-}flow}$ of the section. When the outflow $I_{b,k}(p)$ is lower than the inflow $I_{a,k}(p)$ traffic has accumulated in the section, which indicates congestion. The delay is computed as the time needed for the extra vehicles to leave the section with an effective capacity $C_{eff,k}(p)$ equal to the average outflow of three time periods. This is added to the free-flow travel time to obtain an estimated travel time. This principle is expressed in the equation (8.4) to (8.6) and illustrated in Figure 8.3.

$$SLTT_k^{mass\ balance}(p) = t_k^{free\text{-}flow} + \frac{N_k(p)}{Ce_{ff,k}(p)} \tag{8.4}$$

$$C_{eff,k}(p) = \frac{1}{3}(I_{b,k}(p) + I_{b,k}(p-1) + I_{b,k}(p-2)) \tag{8.5}$$

$$N_k(p) = N_k(p-1) + I_{a,k}(p - t_k^{free-flow}) - I_{b,k}(p) \tag{8.6}$$

where $N_k(p)$ is the number of vehicles present on k during p, $C_{eff,k}(p)$ the effective outflow out of section k during p and $t_k^{free\text{-}flow}$ the free-flow travel time of section k.

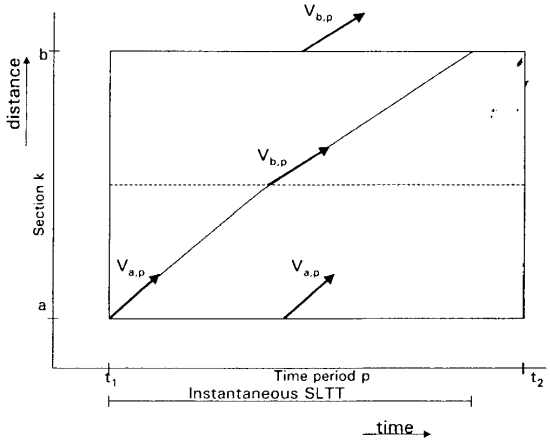


Figure 8.2: Time distance diagram with the principle of the speed-based instantaneous SLTT [thijs00].

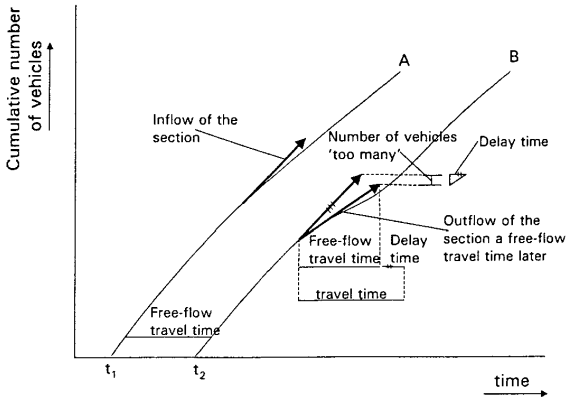


Figure 8.3: Cumulative flow diagram with the principle of mass-balance [thijs00].

The section level travel times can be used to calculate the network level travel time (NLTT; the travel time of a certain route). Also two methods that can be used to calculate the NLTT are discussed: the instantaneous NLTT estimator and the 'weighted' instantaneous NLTT estimator.

The instantaneous network level travel time based on section-level travel times computes the route travel time that would be experienced if all considered sections were to be traversed in the current time period. The method does not take into account that traffic conditions vary over time. The NLTT is summed over all sections of route R:

$$NLTT_R^{inst}(p) = \sum_{k \in R} SLTT_k(p) \tag{8.7}$$

In figure 8.4 this is shown. $V(k,p)$ represents the average section speed defined as the $SLTT_k(p)$ divided by the section length L_k. The average section speeds are indicated in the middle of the (k,p)-region whereas the trajectory indicating the average route travel time starts at the beginning of starting period p.

The weighted instantaneous network-level travel time is weighted by a section-specific weighting factor. The weighting factor is the total distance driven by all vehicles present on that section compared to the total distance driven of all vehicles at the total route:

$$NLTT_R^{weighted\ inst}(p) = \frac{L * \sum_{k \in R} I_k(p) \cdot SLTT_k(p)}{\sum_{k \in R} L_k * I_k(p)} \quad \text{and } I_k(p) = I_{b,k}(p) \tag{8.8}$$

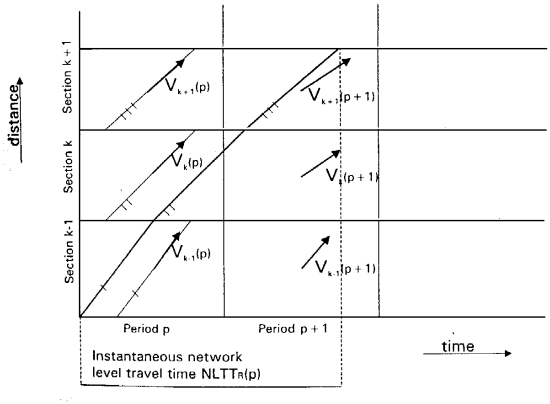


Figure 8.4: Derivation of the instantaneous NLTT from SLTT's [thijs00].

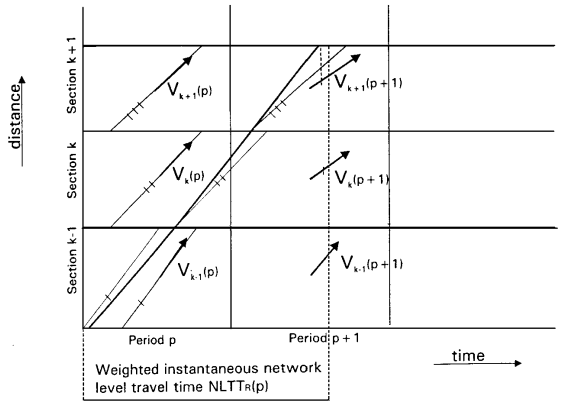


Figure 8.5: Derivation of the weighted instantaneous NLTT from SLTT's [thijs00].

In Figure 8.5 each section has been given a factor α_k representing the distance driven on that section in that period relative to the distance driven on the whole route in that time period. The average speed assumed that for this section with the weighted algorithm is therefore not parallel to the measured average speed represented by the bold arrows.

8.4 Travel time prediction

The methods discussed in the previous section only estimate *current* travel times. When choosing between different alternative routes this is not the information needed to make the correct decision. One needs to know what the travel times at the moment of traveling along each alternative route are. Of course the current travel times somehow indicate what the future travel times will be, but a better prediction would enable a better choice.
Fortunately, a lot of research is being done in the field of travel time prediction. Unfortunately the results of this research are not yet as good as they need to be. Especially when the prediction horizon is further away (> 15 - 30 minutes) no better prediction then the retrieval of historical data can be made. In [lint00] it was stated that "In all fairness, the historic travel time estimator also performs quite well, indicating that one day looks almost like another." It should be noticed that this is meant for days of different weeks, not for days of the same week. Since route planning mostly requires a time horizon of more then 15 minutes, the fact predictors are outperformed by historical data is a serious problem.
In this section an introduction is given into different prediction techniques and the data they are based on. In the next section use of historical data is discussed.

Statistical traffic model.
For traffic-time prediction a state-space model of traffic on the network is used. Examples are the STM (grol97) and METANET (kots97). During each time interval the traffic state is assumed to be constant. The variables comprising the traffic state model are: density, flow and speed. The state-space model consists of observation equations (which relate the traffic state variables to the measurements), and transition equations (which describe the evolution of the traffic system).

Using the continuous stream of 1-minute traffic measurements, the current traffic state on the network is updated in real-time on basis of observation equations. Starting from the most recently estimated traffic state, the transition equations are used to forecast the future state traffic situation for a rolling horizon of 20 minutes. This gives predictions for section-based traffic speed, flow and density. From the section-based traffic speeds, dynamic network-level times are calculated.

Behavioral traffic model & OD (Origin Destination) demand estimation prediction.
Another approach is to use dynamic assignment models to derive travel times. MIDA is such a dynamic traffic assignment model. MIDA stands for MultIclass Dynamic Assignment. Currently two classes are recognized: normal cars (70-80%, which includes small business vehicles) and the remainder, heavy vehicles. MIDA cooperates with other modules, which are an input generator, and an OD demand estimator and predictor respectively.
The OD demand estimator provides the assignment model with the best estimates of the OD-matrices of past periods, based on a historical database, but more importantly on the basis of current measurements. The OD prediction model is based on a filtering approach, which combines historical and estimated OD information with predicted inflows at the highway ramps. Using the estimated and predicted traffic demand, the time varying traffic conditions are determined for all road-sections during a certain simulation period (one hour).
Apart from MIDA, many other dynamic assignment approaches are available that can serve the same purpose.

Instantaneous travel time based on predicted flows.
The instantaneous travel time mode based on predicted flows consists of three basic steps:

- Prediction of the traffic load on a route between an origin and destination,
- allocating the total traffic to the sections of the route,
- estimating the speed on each section to calculate the travel time.

The traffic load is the distance covered by all vehicles on all section in a time period. It is estimated on the basis of recent historical traffic data by applying a regression model, which includes spatial and temporal explanatory variables.
The second step precisely allocates this load to the various sections and thus gives an estimate of the flows on each section for the following time period. The used allocation is the distribution of vehicles that is observed at the last measurement.
The last step uses these flows in order to estimate the speeds on each section on the basis of speed-flow curves. Finally, the NLTT will be estimated based on the calculated speeds for each section.

Neural networks.
Neural networks seem promising in the field of travel time prediction. In [lint00] an overview of applications of artificial neural networks (ANN's) for travel time prediction reported in the literature is given. It is stated that only few applications of ANN's have been found, although the characteristics of travel time prediction would suggest a neural network approach would be very obvious. The results of the few applications show promising results, both in the field of indirect travel time estimation, where ANN's are used to predict speeds, intensities, etc. that are used to calculate the travel time, and in the field of direct travel time estimation, where the travel times are the (direct) output of the ANN. In indirect travel time estimation methods like the ones described in the previous section can be used to calculate the travel times. For predictions of more than 30 minutes ahead, using the historical profile gives better results than the other approaches.
It was also noticed that most attempts to predict travel time are designed on the basis of only one or two measured quantities, from one data source, like inductive loops, automatic vehicle

identification systems, etc., while much more inputs are available. In [lint00] it is argued that "much can be gained by consolidating different types of information using neural data fusion techniques, that is, by considering (input-) data from different sources and feeding these to the neural network. By the means of training, the data fusion system is to decide which data (inductive loops, video, AVI, probes, historic data) can best be used under the prevailing circumstance, given a specific prediction horizon." In Figure 8.6 an overview is given of factors that influence the individual travel time and that might be used as input.

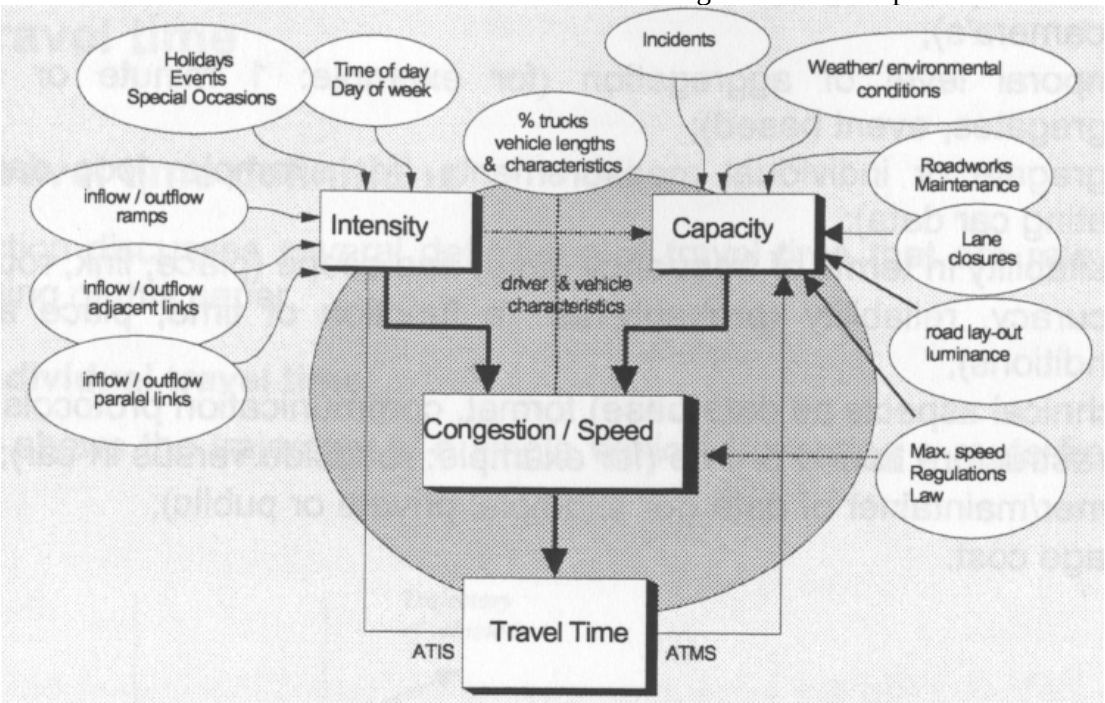


Figure 8.6: Factors that influence the individual travel time [lint00].

8.5 Historical data

As was stated in the previous section travel time predictors are (almost) outperformed by historical data, especially when the prediction horizon becomes further away. Apparently the load on the freeway network is almost the same on compatible days (same day of the week). Neural networks show promising preliminary results, but final results are not yet available. Consequently, one might suggest to (simply) use historical data for the travel time estimation module in KRIS.
Unfortunately, this approach has one considerable drawback: when unexpected events, like accidents, happen a travel time prediction based on historical data will be completely wrong. Somehow, the actual state of the network has to be taken into account, especially when predicting travel times in the near feature (< 15 minutes). When predicting travel times that are further away in the future, historical travel times could be used. In between some smooth averaging procedure would have to be applied, to tune both values.

8.6 Data used by KRIS

After the previous sections it may be clear still a lot of research has to be done into the accurate prediction of travel times. As was stated before this was outside the scope of this graduation thesis. Fortunately, a lot of research is being done into the development of travel time predictors [bovy00]. As stated before, it can be concluded that the possibilities of KRIS are greatly determined by the quality of the travel time estimation module. There are two

ways to provide KRIS these data, via data from the MONICA system and using the positions of the KRIS users. Somehow these data have to be translated into travel time predictions.
At the moment time no prediction module has been developed yet. Since a real-time performance of the system was not a goal of this research, this was not a insurmountable problem. The correct operation of the KRIS can also be shown using historical data, since the goal of the research was to investigate the feasibility of a system that is able to find alternative routes, using dynamic and historical data. Consequently a data set had to be constructed, that would be good enough to show the potential of the KRIS. To be able to study the performance of KRIS, especially considering issues like robustness, these data should be realistically enough. Consequently historical data were used that are based on measurements of the MONICA system in 1999. This section will first introduce the way a data set of historical data was created. Thereupon it is described how dynamic data was constructed.
A simple approach was chosen to prevent too much time of this graduation work was spend in generating a data-set, since the data are only needed to illustrate the correct operation of KRIS. A map of the motorway network of the Netherlands was used, with for each hour of the day the congestion probability based on measurements in 1999. In Figures 8.7 and 8.8 two of these maps for different hours of the day are shown. Using these maps travel times

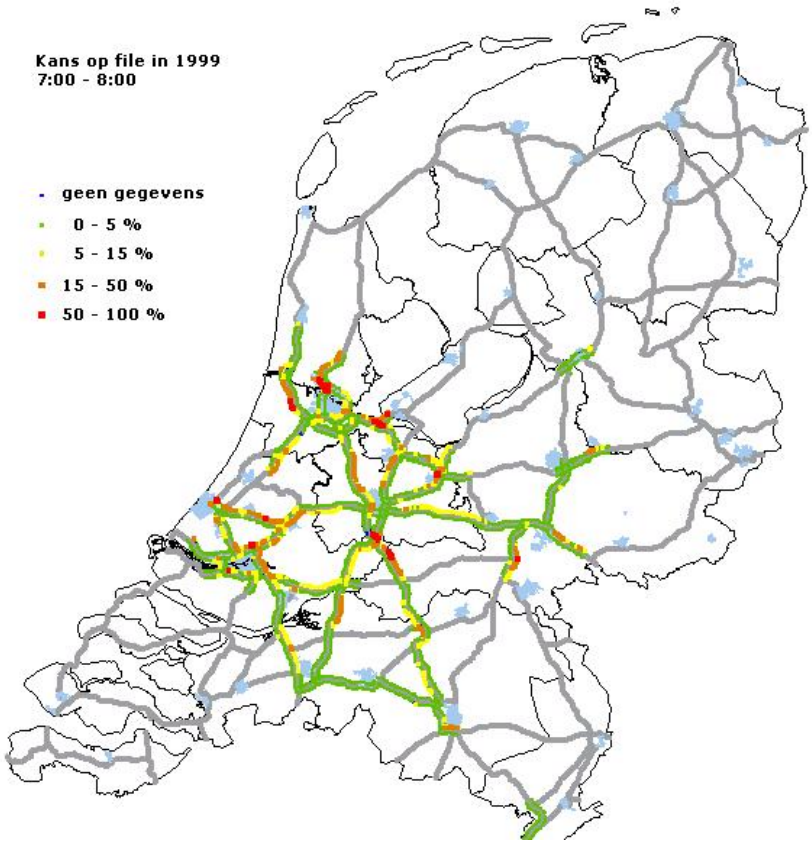


Figure 8.7: Congestion probabilities between 7 and 8 a.m..

were estimated. First the travel times in free-flow traffic were estimated, using special measurements along the A2, A4, A9 and A10 and the results of different route planners available at the Internet (the results of the different route planners were compared to the measurements along these roads and the best one was chosen). According to the probability of congestion for a certain hour of a day this estimated travel time was adjusted. This process of estimation was made by a rough calculation using fuzzy rules like ‘if the chance is >50% then double the travel time’. Of course this is not a very accurate way of constructing historical data, but that is also not the goal of the research. As said before, the data is only

needed to illustrate the correct operation of KRIS. The day was divided into four parts: the morning rush hour (6 a.m. to 9 a.m.), the daily hours (9 a.m. to 15 p.m.), the evening rush hour (15 p.m. to 19 p.m.) and the night (19 p.m. to 6 a.m.). For each motorway a file was constructed with for every part of the day an estimated travel time from each ramp or junction to the next ramp or junction. These files can be seen as three dimensional matrix of the starting point of a section, the end point of a section and the time of the day: an origin-destination matrix with different layers for different times of the day. An example of such a file is shown in appendix D.
Now data was available to plan routes in the future. As was stated in the previous section, also up-to-date information should be incorporated, since unexpected events may influence the travel time significantly. When an accident happens often very long delays arise along only one freeway. Such accidents can be used very well to demonstrate the route planning capacities of KRIS.
It was chosen to let the user be able to generate such unexpected queues. A separate module has been developed in which queues can be created by setting the travel time between two ramps. In section 9.4 this is illustrated.
To make KRIS easy adjustable in the case real-time data becomes available, the design of the system was made in such a way that this could easily be done. As was already stated in the requirements (chapter 4) the design of KRIS had to be modular. KRIS uses a method 'GiveEstimatedTravelTime' that simply returns the estimated travel time from a ramp or junction at a motorway to the next ramp or junction. At the moment this method is implemented by using the data constructed in the way described previously, but when real-time data and a good predictor become available the only part that needs to be changed will be this method. This way it is quite easy to couple KRIS to another source of travel time predictions, whether this may be a specific predictor, data directly from MONIBAS or data derived from user positions.

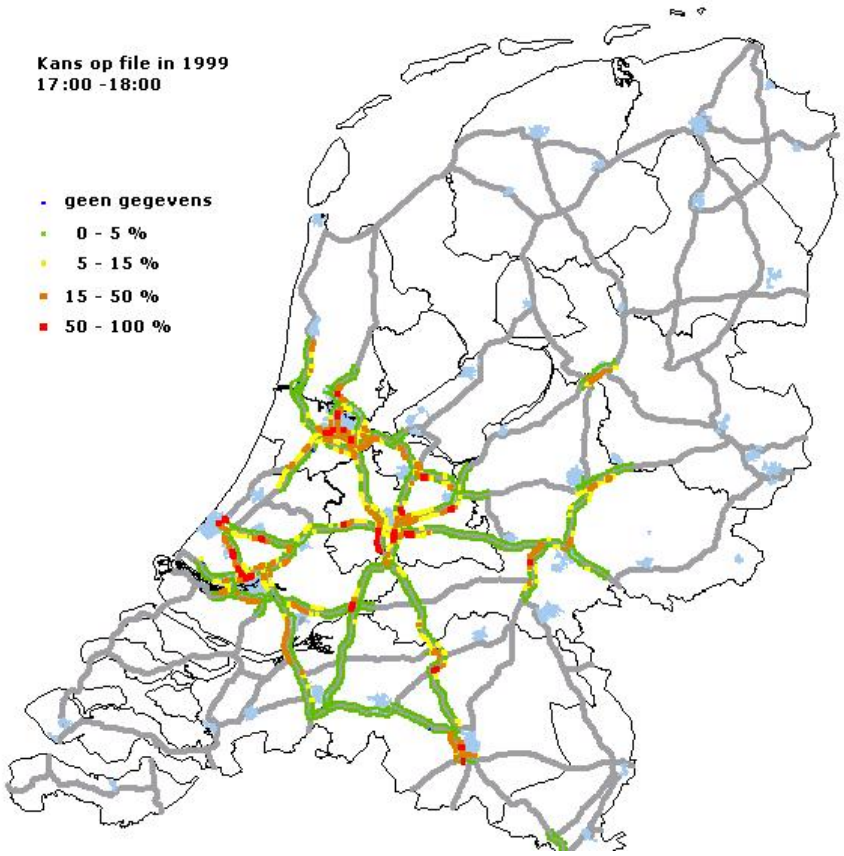


Figure 8.8: File chances between 5 and 6 p.m..

PART III:

RESULTS

Chapter 9: Implementation

In this chapter the implementation of KRIS will be discussed. Firstly, in section 9.1 an introduction to the implementation will be given followed by a discussion of the OO (Object Oriented) model that has been made (section 9.2). In section 9.3 a detailed description is given how information is retrieved from different websites. In section 9.4 an introduction to the program itself is given, while in section 9.5 the prototype that has been implemented is evaluated.

9.1 Introduction

In part II of this thesis the design of KRIS has been reviewed thoroughly. In chapter 5 the design of the algorithm that finds the best multi modal route using dynamic data was introduced, while in chapters 6 and 7 the designs of the expert system and extended Dijkstra algorithm for finding the best dynamic car route were introduced. Both approaches are used to find the car route parts of a multi modal route. In chapters 6 and 7 also some remarks were made about the implementation of both approaches (sections 6.8 and 7.5). These implementations are used by the algorithm that was described in chapter 5 to find the optimal car route. In this chapter the implementation of this design using both expert system and extended Dijkstra algorithm implementations will be discussed.
It will be clear that KRIS has to process a lot of information. In the requirements that were given in chapter 4 it was stated that the design should be modular, so different data sources could be exchanged easily. Consequently the implementation had to be done in such a way that the total functioning of KRIS did not depend on one or more data sources. Ideally, KRIS can choose between data sources. An example is the availability of two implementations to find the shortest car route (graph and expert system), which both can be used by KRIS. Considering these implementation issues an object oriented model was created. A lot of effort was invested in the design of the OO model, since this would ease the programming task significantly and would guarantee a modular design wherein data sources could be exchanged easily. In section 9.2 the OO model will be discussed.
The OO model was implemented using the Microsoft Visual Java environment. Of course, also other (OO-)languages could have been used, like C++ or Delphi. The major advantage of the Java programming language was (and still is) that a lot of extensions (so called packages) are available. As was stated in section 6.8 the expert system package, Jess, is a very useful expert system extension, especially since CLIPS rules do not have to be modified. Secondly a package was available that implements a major part of the extended Dijkstra algorithm: classes of the nodes and links are available together with a framework in which the algorithm can be implemented. The availability of such packages was critical in choosing the language. In section 9.3 an interesting detail of the implementation is shown: the way KRIS retrieves information from the different websites. The information retrieval from different data sources formed a significant part of the total implementation and is essential, since without appropriate data KRIS can not function.

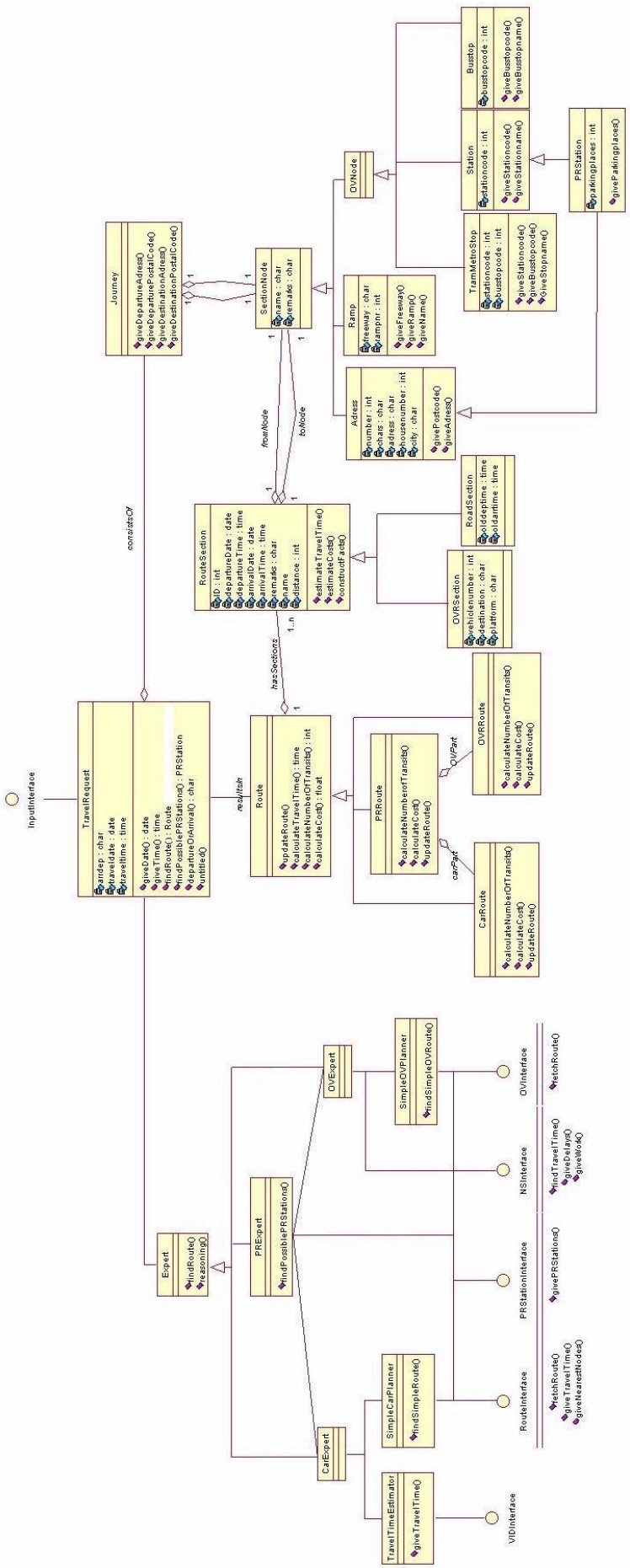


Figure 9.1: Class diagram of KRIS.

9.2 OO Model

In Figure 9.1 the class diagram of the OO model is shown. In the right part the classes that store a car, public transport or P+R route can be found: a Route consists of RouteSections that each have two SectionNodes. These SectionNodes can be of different types, for example an Address, Ramp, Station, BusStop, etc., while the RouteSections can be of a car route (RoadSection) or of a public transport route (OVSection). Finally, a Route can be a CarRoute, an OVRoute or a PRRoute that consists of the former two.
In Figure 9.1 the different classes that implement the algorithm that was presented in chapter 5 are shown. The CarExpert, OVExpert and PRExpert each take care of the construction of their routes: the CarExpert uses the classes TravelTimeEstimator and SimpleCarPlanner. The latter returns a static route, using the RouteInterface class that communicates with a route planner on the Internet (see section 9.3), while the TravelTimeEstimator returns travel times when two ramps are given. The same architecture can be found for the OVExpert that uses the SimpleOVPlanner to fetch a static route (in the next section this planner is described in detail) and an interface to get delay data. Finally, the PRExpert takes care of finding PRRoutes, using the previously mentioned classes.
The implementations of the expert system and the extended Dijkstra algorithm can both be found in the CarExpert class, although for the latter special classes were used, since the JDSL package was used (see chapter 7). This is shown in Figure 9.2, where the ShortestPath and JunctionGraph classes and their relationship are shown. The expert system completely resides in the CarExpert class, in the method reasoning. A Boolean variable determines whether the method findRoute (of the CarExpert class) will invoke the expert system in this method or will use the extended Dijkstra algorithm implementation and its classes.

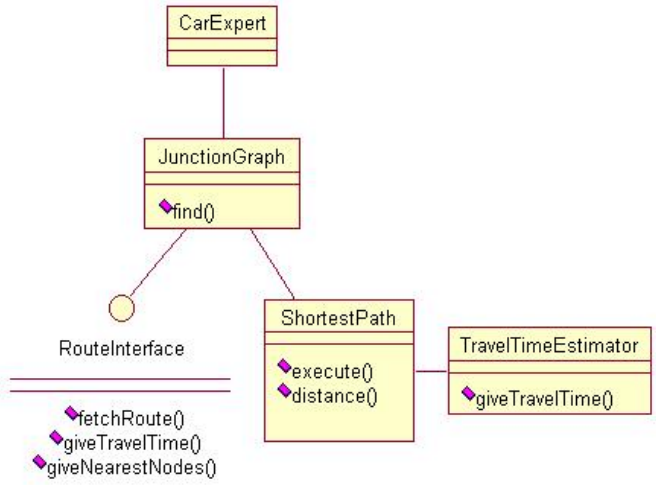


Figure 9.2: Class diagram of the extended Dijkstra Algorithm.

Apart from the class diagrams also interaction diagrams were developed, to capture the communication between the different classes and the way the algorithm is implemented. In Figure 9.3 the interaction diagram for the extended Dijkstra algorithm is shown. The route planner constructs a static route of which the entrance and exit ramp are taken (giveNearestNodes(pc,pc)), which are used to find the shortest path between these nodes. To compute this shortest path traveltimes have to be computed for all trajectories that are investigated. Finally the route is returned as a string, of which a CarRoute object is constructed. The other interaction diagrams can be found in Appendix A.

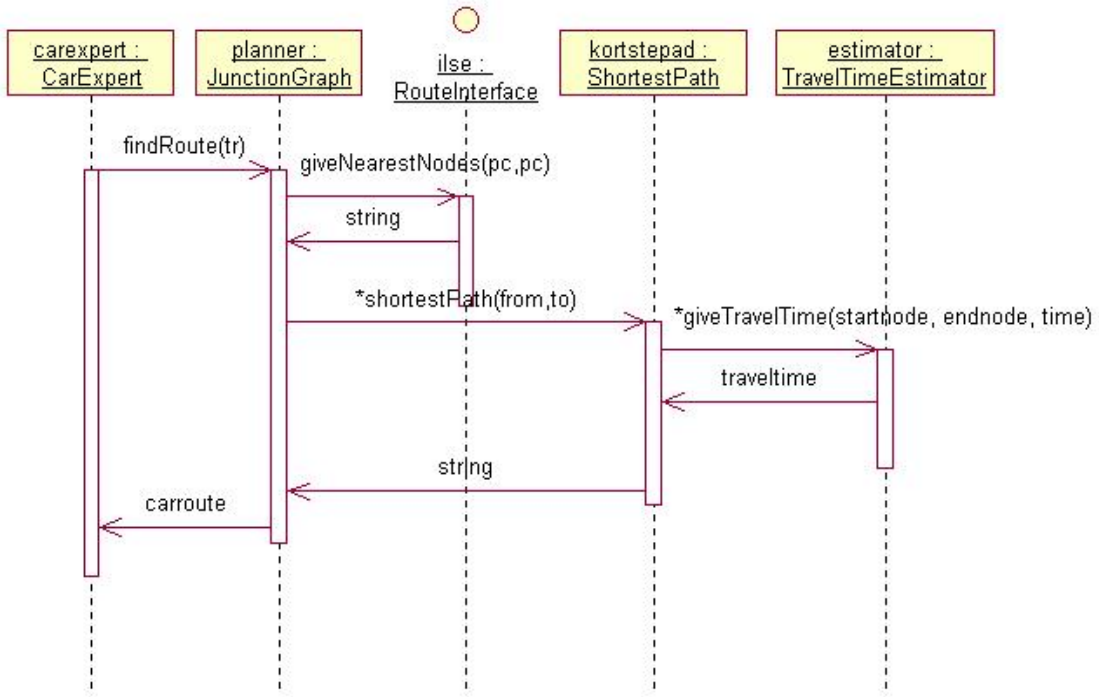


Figure 9.3: Interaction diagram of the extended Dijkstra algorithm.

9.3 Retrieving information

The most complex implementation task was to implement the modules that retrieved the needed information from the appropriate websites. The postal code website (http://www.postcode.nl) was used to be able to translate postal codes in addresses and vice versa, the Ilse route planner site (http://kaart.ilse.nl) was used to retrieve static car routes and the public transport site (http://www.9292ov.nl) was used to find public transport routes. Sophisticated coding was needed to be able to retrieve the needed information. It has to be clear what information should be send to the site in order to retrieve the information needed: the exact request string has to be constructed in the right order. When this string has been send to the site, HTML (HyperText Markup Language) code is returned. The information needed has to be inferred from this HTML code. Finding the information in such a HTML coded page is called parsing. This process is illustrated in this section by showing the design scheme and Java code that was constructed to be able to use the public transport website.
In Figure 9.1 a schematic overview is given of the needed communication between KRIS and the public transport site. It took quite some time to discover what data had to be sent to the site in order to receive the needed information. Firstly a session ID is needed. This identification has to be used in every request that is send to the public transport site, since otherwise it does return the start page. To be able to send a route request x, y co-ordinates are needed of the departure and destination address. Normally these co-ordinates are not shown to a user that simply uses the website, since they are only used internally. A smart request was made to obtain these co-ordinates. Using these co-ordinates the final route request can be done. The public transport website returns the best routes in a HTML coded page, which has to be parsed to obtain the needed information.
In Figure 9.5 the code is shown that retrieves the session ID number. First the website and the query that has to be send are instantiated (URL and action), while in the second try block, the website is contacted and the received string is being decoded, by finding the string "<form ACTION=". Behind this string the id can be found.

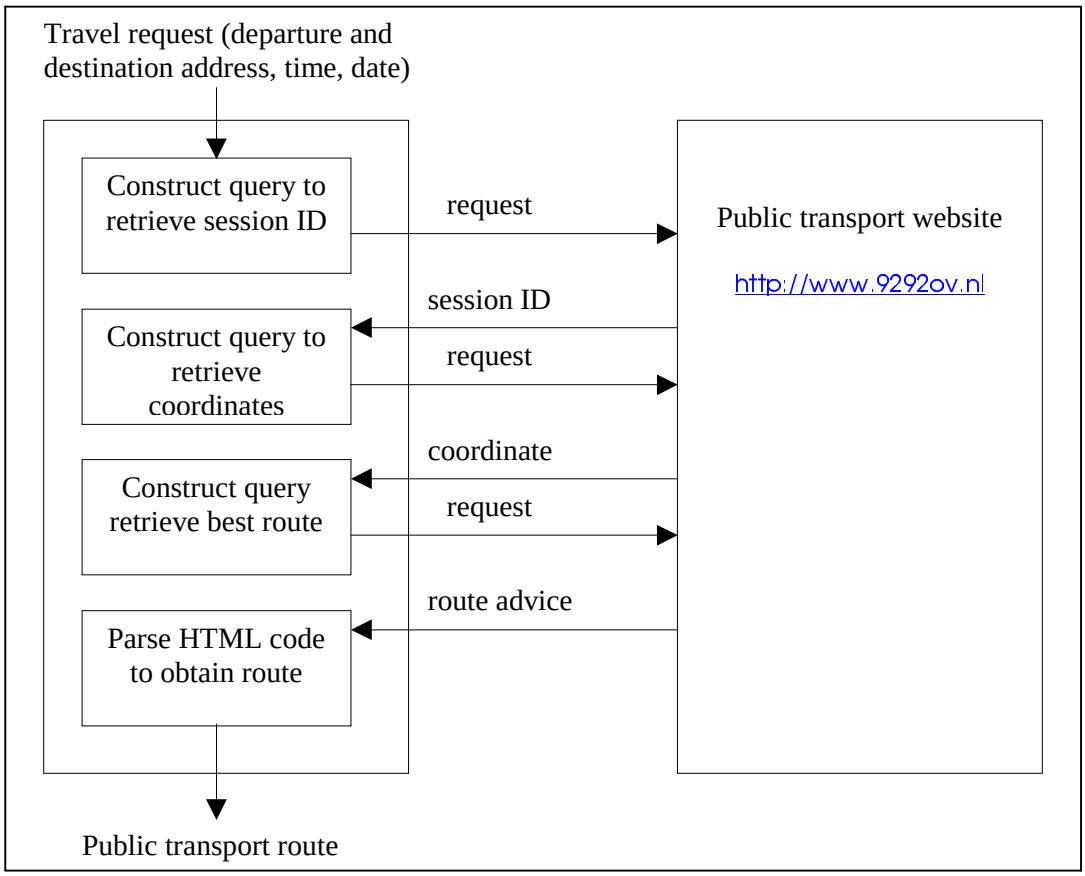


Figure 9.4: Schematic overview of communication with public transport website

```
try {
        URL url = new URL("http://www.9292ov.nl");
        action = "/main/redirect.asp?ID=47";
        actionURL = new URL(url, action);
} catch (java.net.MalformedURLException e) {
        Form1.displayError("Couldn't contact http://www.9292ov.nl");
}
try {
        connection = actionURL.openConnection();
        connection.setDoOutput(true);
        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String site = in.readLine();
        while (site!=null)
        {
                w.write(site);
                if (site.startsWith("<form ACTION="))
                {
                        try {
                                gnu.regexp.RE regexp = new gnu.regexp.RE("&id=\\d+");
                                gnu.regexp.REMatch match = regexp.getMatch(site);
                                id=site.substring(match.getStartIndex()+4,match.getEndIndex());
                        } catch (gnu.regexp.REException e) {  }

                }
                site=in.readLine();
        }
} catch (java.io.IOException e) {
        String out = e.toString();
}
```

Figure 9.5: Java code that retrieves the session id from the public transport site.

The same way the other requests are send, although each request has its own action string containing the string that should be provided (in Figure 9.5 the action string was "/main/redirect.asp?ID=47"). In Figure 9.6 the action string is given to retrieve the x and y co-ordinates. For city1, street1, city2 and string2 the departure and destination address are filled in. In Figure 9.7 the string that is returned containing these co-ordinates is illustrated. Using the same methods the actual route advice is retrieved.

```
action = "/9292/asp/MaakVraag.asp?PVan="+city1+"&SVan="+street1+"&NVan=&PNar="+city2+"&SNar="
street2+"&Nnar=&PVia=&SVia=&NVia=&dtm=11-6-2001+9:21:48&VeAa=vertrek&toha=&viad=&id="+id;

actionURL = new URL(url, action);
```

Figure 9.6: Request string to retrieve the co-ordinates of the departure and destination address.

```
"PVan=delft&SVan=zuidplantsoen&NVan=&PNar=dordrecht&SNar=johanna+naber+erf&NNar=&PVia=&SVi
a=&NVia=&RVanX=8523&RVanY=44684&RNarX=10997&RNarY=42387&RViaX=RViaY=1&VeAa=vertre
k&ADSL=D&dtm=2001-4-23+14:26&NrSl=-1&toha=&viad=&MIS=J"
```

Figure 9.7: Received string containing the departure and destination address.

9.4 Illustration of KRIS

In this section an illustration will be given of the working of KRIS. This will be done following the example that was also used in section 5.6. In this example someone wants to make a journey from the village Amerongen to Amsterdam. In Figure 5.9 both cities and the railway and road networks between them were shown. The traveller wants to leave at 8 a.m. and is curious about the best route to take. To illustrate the working of KRIS first a travel

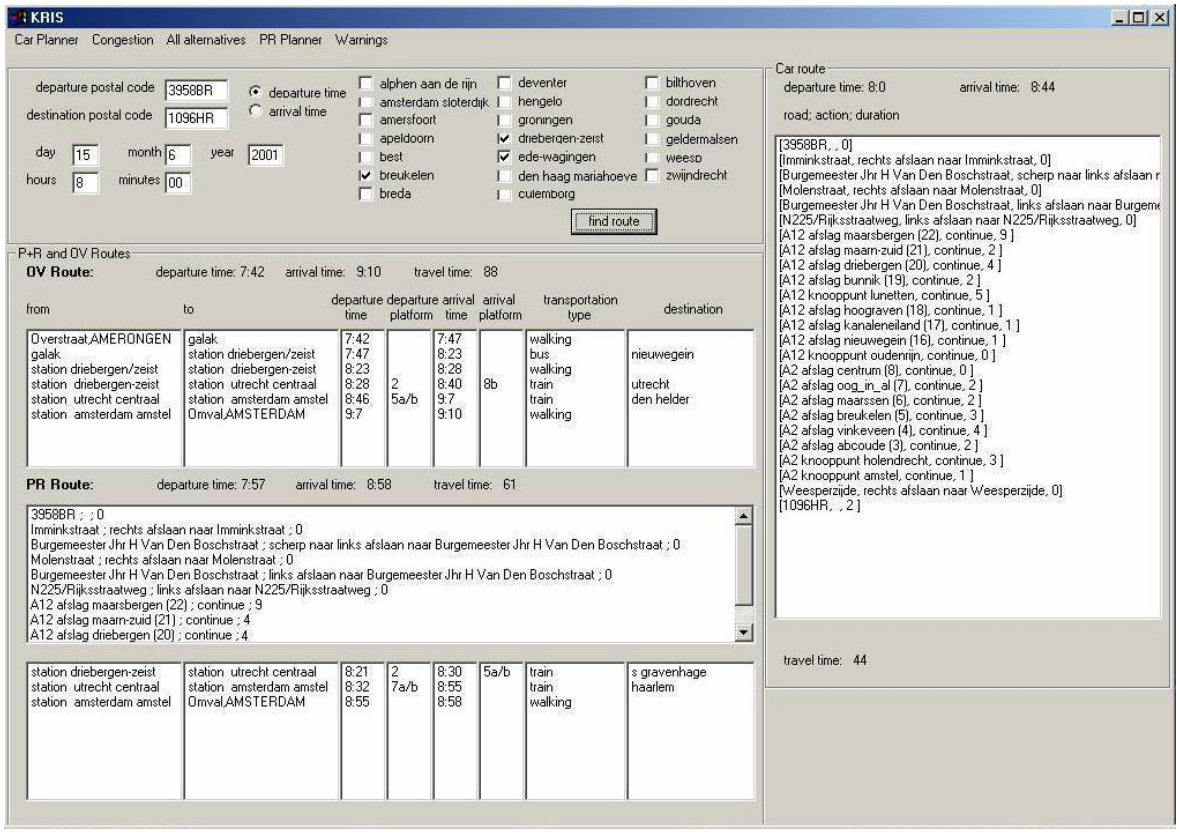


Figure 9.1: The result of a route request under free flow conditions.

advice is given considering free-flow road network conditions: no historical data are used yet. Then delays along the car route that has been chosen are incorporated and a new travel request is done. In Figure 9.1 a screen shot of the result of the request in free-flow conditions is given.
In the most right part of the screen the fastest car route is shown. Below the route details the total travel time is shown, 44 minutes. In the left part of the screen, in the P+R and OV (public transport) routes-section first the best public transport route is shown. As can be seen above the details of the route the total travel time of this route is 88 minutes. Finally, below the public transport route the P+R route is shown, consisting of a car route part and a public transport part. The travel time of the P+R route is 65 minutes.

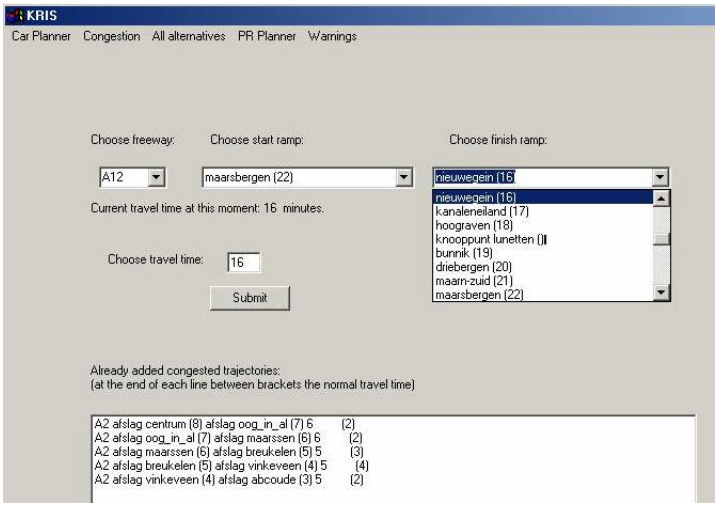


Figure 9.2: Creating congestion along the A2 and A12.

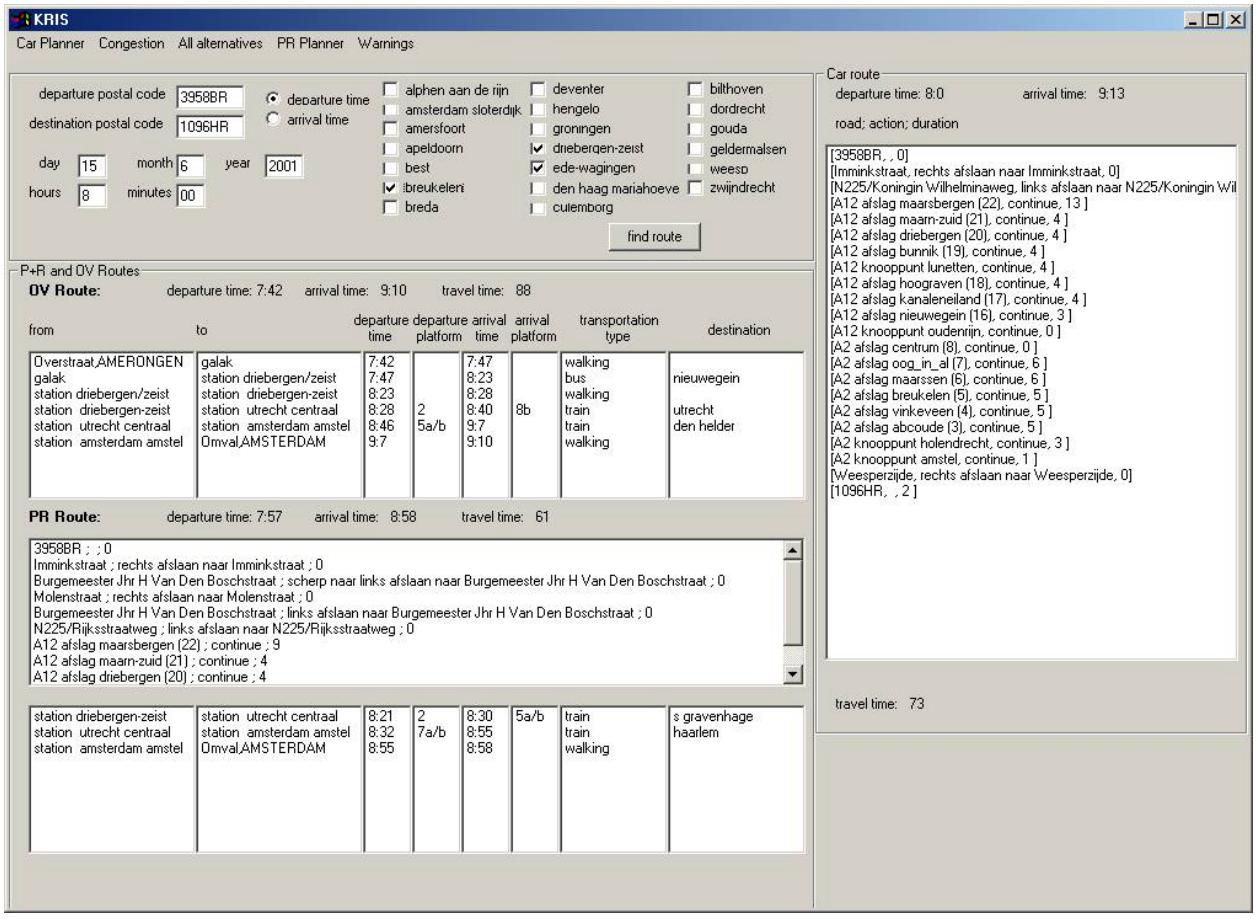


Figure 9.3: Resulting routes considering historical data.

As was stated before, these results were calculated considering a free flow road network. In Figure 9.2 an illustration is given how congestion can be created. Along the A12 a delay of 11 minutes is created and along the A2 a delay of 14 minutes (see also section 5.6). In Figure 9.3 the results are shown when a route request is done incorporating the congestion that was just created. The car route now takes about 73 minutes, so the P+R route is the fastest one now.
In chapters 6 and 7 two implementations of the dynamic route planner for the car were discussed. It was stated that both implementations would be compared. This comparison will be made in the next chapter. In this section the working of both implementations is shown, since that is the subject of this section. Another route is used to illustrate both implementations: the route from Zoetermeer to Muiden. In Figure 9.4 the free flow route and the best route considering all congestion is illustrated. This screen is the result when a route request is done using only the car modality. As can be seen in Figure 9.4 one can choose between the graph algorithm implementation and the expert system. In the corner right under the performance of one of the chosen implementation is shown. In the next chapter more details will be given about the performance criteria that are shown. In Figure 9.4 it is also shown that the free flow route is represented using a red colour (the one that is found by a static route planner), while the best route considering congestion is shown in a green colour.
Figure 9.4 is the result of a route request when the A4 is congested. In Figure 9.2 it was illustrated how congestion can be created in KRIS. In Figure 9.4 it can also be seen that this route was found using the expert system. In Figure 9.5 it is illustrated that KRIS also gives all alternative routes that were tried: when using the “All alternatives” module, this screen is the result, of course only if the expert system was used. In the original and alternative routes box first the static route is stated and below the static route the different alternative routes that were generated by the expert system can be found.More details are also given in chapter 10.

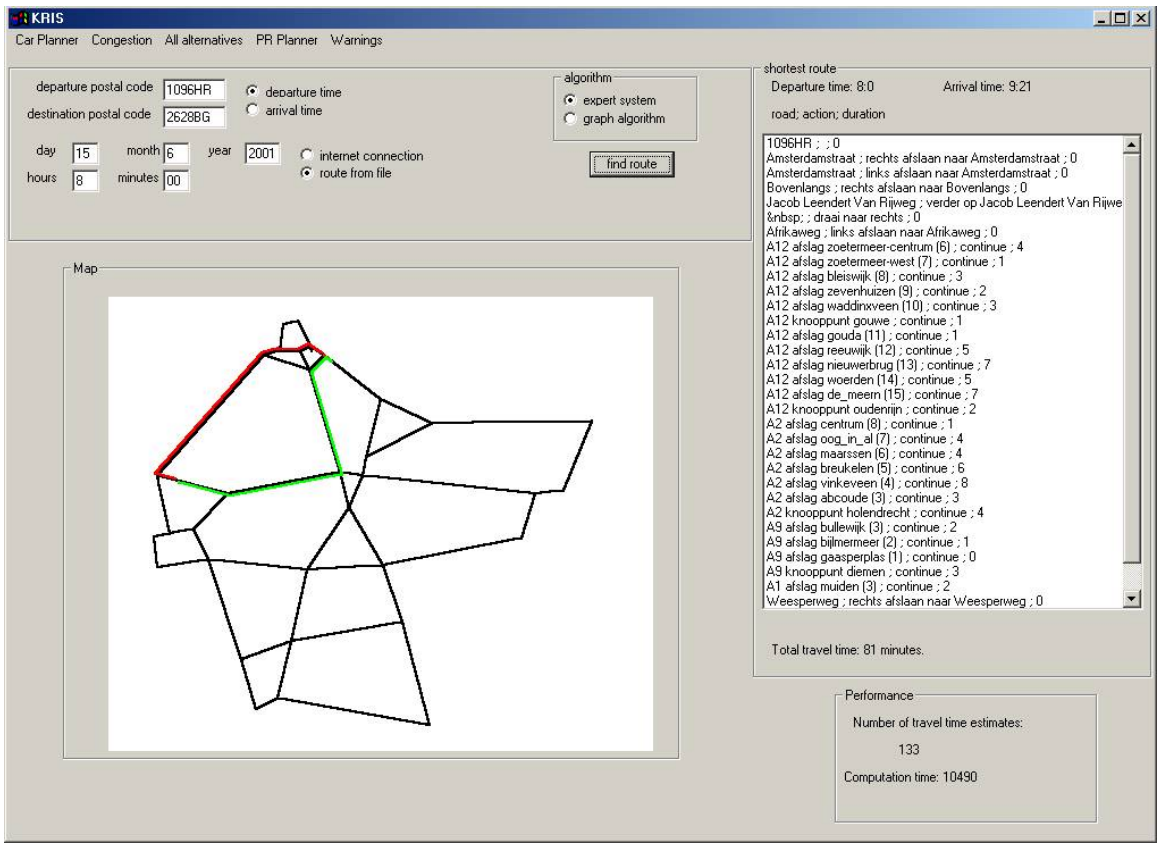


Figure 9.4: The best route and free flow route from Zoetermeer to Muiden.

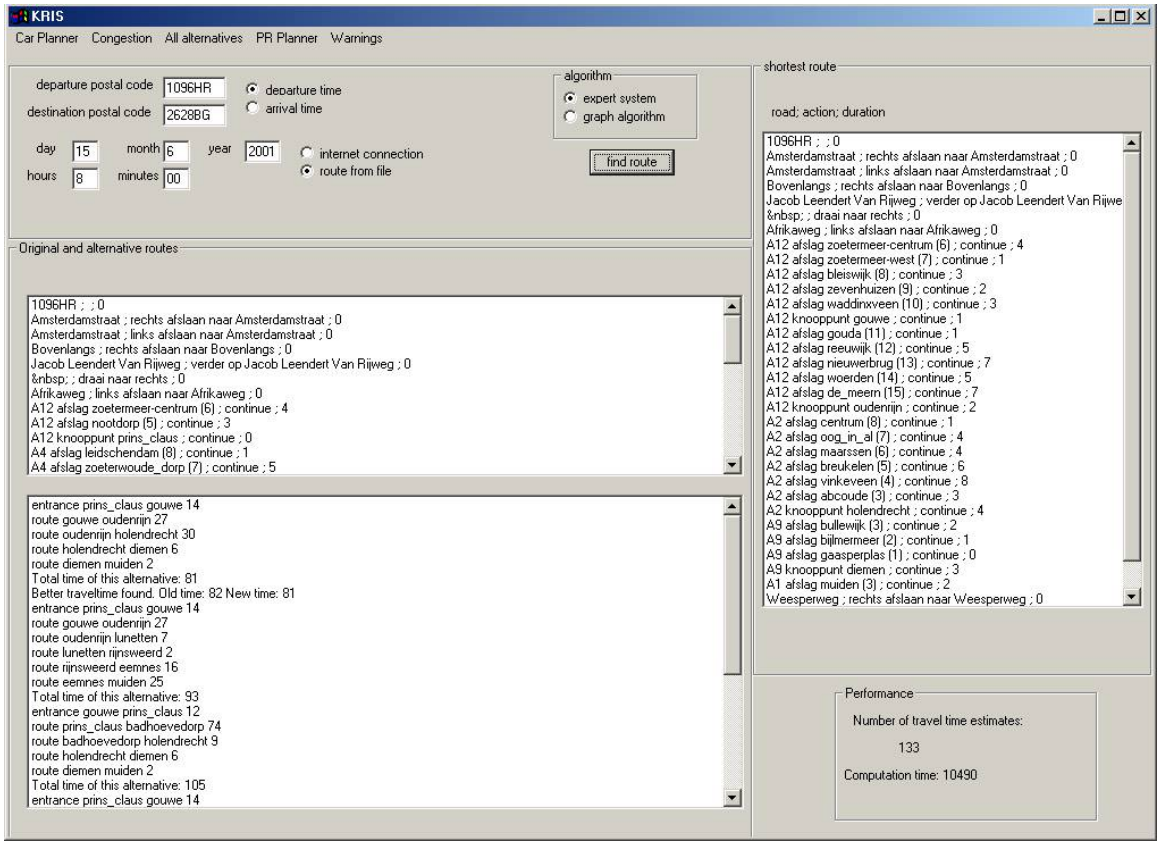


Figure 9.5: The alternative routes the expert system generated.

9.5 Evaluation

In the previous section an illustration of the working of KRIS was given. It was shown that KRIS is capable of finding a multi modal route, especially in the case of congestion along the highways. Several tests were done to see if KRIS functions correctly and thoroughly testing was done to compare the expert system with the extended Dijkstra algorithm. The results of these tests will be discussed in the following two subsections, while the results of the comparison between the expert system and the extended Dijkstra algorithm will be discussed in the next chapter.

9.5.1 Comparing advised multi modal routes with historical data

To illustrate the working of KRIS in the previous section examples were given that started along a free flow network: no historical data was incorporated, although this feature is one of the main strengths of KRIS. As was stated in chapter 8 at the moment prediction models produce only slightly better results then using historical data, which KRIS actually does.
To assess the quality of the routes that KRIS advises testing was done by performing several route requests to KRIS from different departure to destination addresses (see Figure 9.1). Using the historical data of the year 1999 the routes KRIS found were assessed: it was investigated if the routes actually were the fastest ones for that moment of the day. Indeed the routes KRIS returns are the best one considering the data set of the year 1999 and the time tables of the different types of public transport. This is not a very big surprise, since the data KRIS uses to find the best route are also based on the historical data of the year 1999.

9.5.2 Robustness of KRIS

To test the robustness of KRIS it was tested under which conditions the route advice changed to another route. Since small differences in travel times may result in totally different route advises this is an important issue to address. For example, if two routes have almost the same travel time, but travel in opposite directions along the freeway one is following, the route advice might switch constantly between both directions. At the one moment, the advice is turn around and travel in the other direction, while after a slight delay of that route the advice can be exactly opposite. A good travel guidance system would detect such controversial route advises and would only advice another route if the travel time would have changed significantly enough.
To evaluate KRIS's performance for this issue, a route advice was asked for different routes and for each route congestion was created minute by minute. It was monitored when the route advice changed and then congestion along that route was created minute by minute. Again was monitored when the route advice changed, and especially if it changed towards the previous route.
From these tests it can be concluded KRIS changes from one alternative route to the other immediately. For all routes that were tested, it was monitored that immediately when another alternative route was only a minute faster, this alternative route was advised. Actually, this is also what one would suspect, since KRIS optimises the current route request, without taking into account previous route requests. Consequently a filter should be constructed in the PITA that uses KRIS or in another module, to filter these sudden changes, since no traveller would take KRIS's advice seriously if it changes constantly between two opposite directions.

9.5.3 Evaluation of the requirements

It can be noticed that KRIS meets not all of the requirements that were stated in chapter 4. For example, KRIS only returns the shortest route in travel time and does not allow another definitions of the term best route. Another example is the fact that KRIS only takes into account delays along the highway network: no train, bus or other public transport delays are incorporated, which may influence a journey significantly.
Nevertheless it can be stated KRIS functions successfully. In chapter 1 the goals of this graduation research were stated. One of the goals was the development of a prototype that could illustrate the feasibility of a multi modal route planner using this design. Consequently some requirements were met, while others will have to be fulfilled during further development into a fully functional system. In chapter 11 a detailed discussion can be found about the extent in which these goals were met. In the remainder of this section an overview of the functionality of the prototype with respect to the requirements that were stated in chapter 4 is given. For some requirements a detailed discussion about a different performance due to a different implementation (expert system and extended Dijkstra algorithm) can be found in chapter 10.

Incremental searching: When using the expert system searching is done incremental. On the contrary the graph algorithm does not support incremental searching. In chapter 10 this difference is discussed in detail.
Optimal route for an individual: KRIS finds the optimal route for the individual. As was stated in the requirements, nothing is done to optimise the travel time of a group of travellers.
Multi modal routing: Multi modal routing is definitely supported by KRIS.
Dynamic routing: At the moment only dynamic routing is done along the highways. No dynamic data of the public transport are taken into account. It is easy to extend the prototype to a planner incorporating dynamic data of the public transport.
Reliability: When issuing a second route request the same route is advised.

Modularity: The modularity requirement is met very well. It is easy to switch to another data source. During the development the web sites of some information sources changed and it was quite easy to adapt KRIS to these changes. It was also very easy to incorporate two different dynamic routing algorithms to find the best car route.
Scalability: In the case of the expert system approach the scalability requirement is met. When larger networks are taken into account the performance will only increase slightly, due to more rules that have to be examined. The graph algorithm, on the other hand, probably will not be able to meet this requirement. When more nodes will be included the search time will increase significantly (see also chapter 10).
Expandability: It is easy to expand the underlying system. Rules can be added in the case of the expert system and nodes and links can be added in the case of the graph algorithm. It is also very easy to add P+R stations, since they only have to be added to a list.
Understanding of the knowledge representation: The rules of the expert system can be understood very well (see Table 6.1 for an example of the rules).
Communication: At the moment the only communication that takes place is between the user and the prototype. When KRIS should interact with a PITA like system this can be done quite straightforward, because of the modular design.
Learning: At the moment no learning facilities have been implemented.
Best route definition: KRIS is only capable of finding the shortest route, so it is only capable of handling one best route definition. For different best route definitions different expert systems should be developed, based on other objectives.
Personalised routing: At the moment no personalised preferences are taken into account.
Graphical user interface: A graphical user interface (GUI) is available for the car route. Because of the complicated structure of a public transport route and the fact no GIS was incorporated yet no GUI is available for the public transport part of the route.
Explanation facilities: Explanation is given only through a listing of alternative routes that also were tried and the best route in free-flow traffic conditions. Also the travel times of these routes are given, so the traveller can see why other routes were not chosen.
Existing planners should be used: Existing car route and public transport planners are used.
Historical data: The data set that is used by KRIS is based on historical data, although not very precise. Historical data from weekdays to day would result in better predictions (see also chapter 8).
Real-time data: No real-time data are used (only historical data). To illustrate the working of KRIS real-time data can be imitated by the congestion module. Because of the modular structure of KRIS it is quite easy to incorporate real-time data if they become available.

Chapter 10: Expert system versus graph algorithm

In this chapter a comparison will be made between the expert system and the extended Dijkstra algorithm. After an introduction (section 10.1) in section 10.2 the comparison criteria are stated, while in section 10.3 the testing procedure that was used to compare both methods is introduced. In section 10.4 the results of the testing procedure are given. Finally, in section 10.5 the results are discussed.

10.1 Introduction

As was stated in chapter 5 a comparison has been made between the expert system and the extended Dijkstra algorithm. The assumption was made that the extended Dijkstra algorithm would require a lot more computational time, since it has to estimate much more travel times. When KRIS would be used using MONICA data this might become a problem, since travel time prediction algorithms are quite time intensive. Another drawback of the extended Dijkstra algorithm is that it does not support incremental searching. Only one solution is found and that is the best one. The expert system approach supports incremental searching: first the alternative routes are generated and then the travel times are estimated, alternative route by alternative route. After the travel time estimation of each alternative route a 'best route so far' can be stated. Since the order of estimation of the alternative routes is started with the most promising alternative route the first route found is most likely to be the best one (see section 6.6). On the other hand the expert system also has a major drawback. Apart from the fact it requires a lot of time to define the rules in the rule base, it does not generate all possible alternative routes. As was stated in section 6.6 only the alternative routes suggested by the 'experts' are generated. In the situation of many heavily congested roads alternative routes might become the best that no one would have thought of beforehand. The extended Dijkstra algorithm will find these alternative routes, but the expert system will not (unless coded).

10.2 Comparison criteria

To be able to compare both methods some criteria should be chosen which can be used to make the comparison. The following four criteria were chosen which will be discussed in the remainder of this section:

- number of travel time estimates
- overall computation time
- shortest route found
- order of found routes

The number of travel time estimates was chosen since it gives an indication of the performance of the algorithm. Since the travel time estimation is the process that needs the most computational time the number of travel time estimates will strongly indicate the total computational time needed. It should be noticed that the time needed to estimate travel times depends (of course) on the method that is chosen to estimate the travel time. When only a historical time is fetched this process will not take too much time, but when a sophisticated travel time prediction method is chosen the process will take a significant amount of time (see chapter 8). Since the most promising application of KRIS is in combination with the MONICA monitoring system and some kind of predictor, the travel time prediction process can be expected to become the most important one.

The overall computation time criterion is included to be able to judge the performance of the expert system. The overall computation time gives an indication of the performance. It could be possible, the expert system approach needs few travel time estimates, but is very slow itself, since the rule base is very large. Of course, this also depends on the expert system tool that is chosen, so the performance might be optimised by choosing another tool.
The third criterion on which the methods will be compared is the shortest route that is found. Since it is mathematically proven that the extended Dijkstra algorithm will return the shortest route this criterion is only applicable to the expert system. If good rules have been stated always the shortest route should be returned, but (as was mentioned in section 10.1), especially when severe congestion occurs along a large number of roads the expert system might not return the shortest route.
The last criterion, which will be examined carefully, is the order in which alternative routes are given. This aspect is also only applicable to the expert system, since Dijkstra's algorithm does not return any other route then the best one. It should be examined how many alternative routes have to be computed to find the shortest one.

10.3 Testing procedure

In the previous section the criteria on which the two approaches can be compared were identified. In the following sections the values of these criteria that were obtained during extensive testing will be showed and explained. In this section an overview of the testing procedure is given. In section 10.4 an example is of a test run for one route is given, followed by tables showing the results of all tests, together with an explanation of (the numbers in) these tables.
To be able to compare both approaches different route request should be done with different states of the road network. Both approaches should, of course, return the same 'fastest' route for each request and associated road network and a comparison can be made based on the values of the criteria that were introduced in the previous section.
It was chosen to start each testing session with a route request for a road network that is not congested at all. One of the trajectories of the route that is given by KRIS is now delayed and another request is done. KRIS returns another route on which again a trajectory is delayed. This process continues until no further 'interesting' congestion can be created. In Figure 10.1 the steps of the testing procedures are summed. In section 10.4 the procedure will be illustrated.

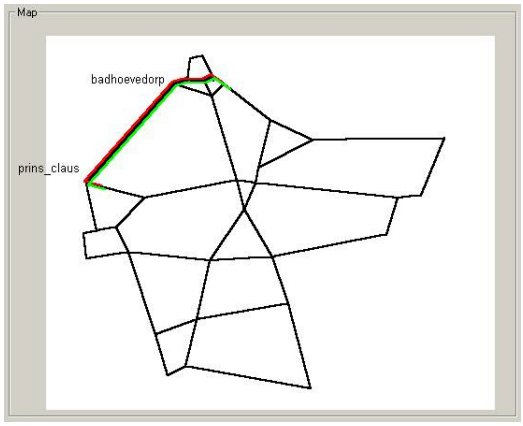
1. Start with a free-flow road network,
2. choose a departure and destination address,
3. make a route request for the departure and destination address,
4. create congestion along one of the trajectories of the route that was returned by KRIS in the previous step,
5. repeat steps 3 and 4 until no interesting congestion can be created any more.

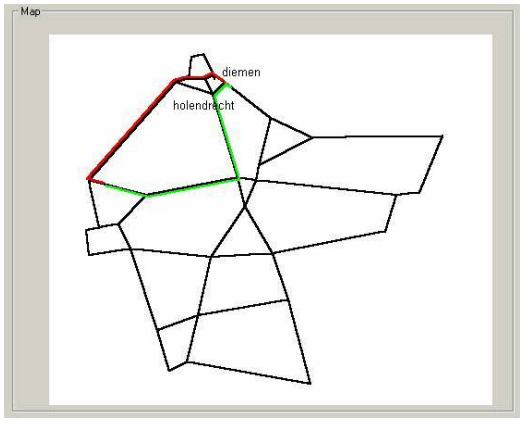
Figure 10.1: The steps of the testing procedure.

10.4 Results

In this section the results of the testing procedure will be given. To illustrate the different steps of the testing procedure will be illustrated with the test results for the route between Zoetermeer and Muiden (between postal codes 2711EE and 1398BK), the same route that was also shown in section 9.2. The route was planned at 20 April 2001 at 12:00, so there was no congestion along the road network. The choice of this time was made, since now it can be observed best which alternative routes are generated in case of congestion along roads that

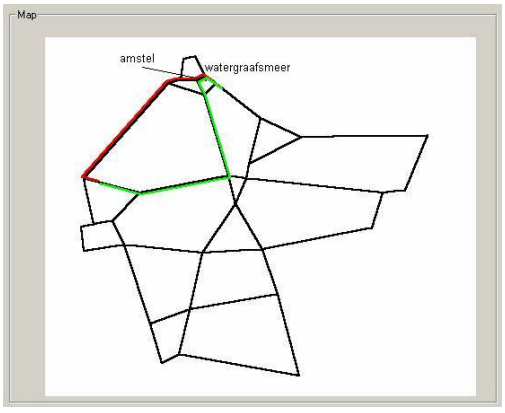
were chosen to be congested. When the route would have started around or in one of the rush hours historical times would be used (see chapter 8) to compute the travel times. These historical times may cause significant delays that might result in different routes. To avoid the problem of identifying delays as a consequence of historical data or as a consequence of a created traffic jam routes are planned outside the rush hours.
Figure 10.2 shows the route that is returned by the expert system, which takes about 48 minutes. The number of route requests was 0. Since there was no delay along any of the trajectories the expert system was not even invoked and consequently no alternative routes were tried. The graph algorithm needed 225 estimates, the computation times were respectively 8620 and 18020 ms.

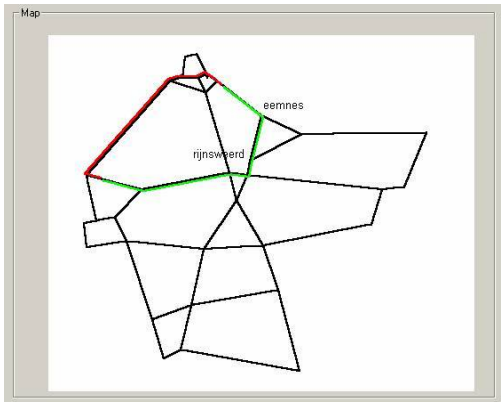




Figures 10.2 and 10.3: The free-flow route (left) and the alternative route when the A4 between the Prins Claus and Badhoevedorp junctions is congested (right).

Now congestion was created along one of the trajectories of the route. Along the trajectory between the Prins Claus and Badhoevedorp junctions 20 minutes delay was created and a new travel request was done. In Figure 10.3 the newly found route is illustrated. The travel time of this new route is 53 minutes. The travel time estimates were 93 for the expert system approach and 1286 for the graph algorithm. The computation times were respectively 16200 and 52340 ms. In Figure 10.3 the junctions between which congestion is created now are also given, the Holendrecht and Diemen junctions: a delay of 15 minutes was created along this trajectory. The new route is shown in Figure 10.4.





Figures 10.4 and 10.5: The route in case of congestion between the Prins Claus and Badhoevedorp junctions and the Holendrecht and Diemen junctions (left) and the route in case of congestion between these junctions and the Amstel and Watergraafsmeer junctions.

The same steps were again carried out for this route. Between the Amstel and Watergraafsmeer junctions along the A10 (see Figure 10.4) a delay of 15 minutes was

created. The resulting route is shown in Figure 10.5. In this route congestion was created between the Rijnsweerd and Eemnes junctions. The route that is now found is illustrated in Figure 10.6. All the results of the different criteria during the described test procedure are shown in Table 10.1.

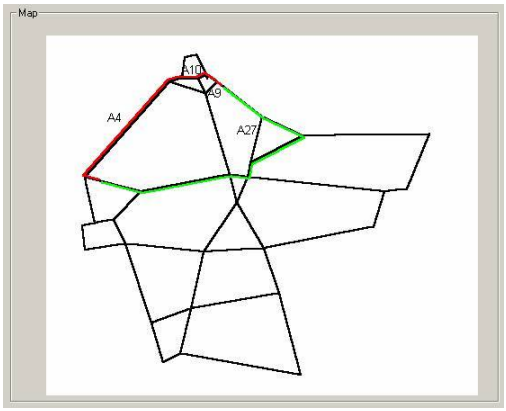


Figure 10.6: Last found route when congestion is created along the A4, A9, A10, A2 and A27.

Table 10.1: Test results of route between Zoetermeer and Muiden. The computation times are in ms.

	exp.syst. trav. est.	exp.syst. comp.time	Graph alg. trav. est.	graph alg. comp.time	order found	Routes are the same
original route, A12-A4-A10-A1 (48'), (Figure 10.2)	0	9500	722	17470	0/0	Yes
delay between Prins Claus and Badhoevedorp (A4), +20', A12-A2-A9-A1 (53') (Figure 10.3)	93	10000	1286	34270	1/3	Yes
delay between Holendrecht and Diemen (A9), +15', A12-A2-A10-A1 (54') (Figure 10.4)	94	10490	1285	34880	3/3	Yes
delay between Amstel and Watergraafsmeer (A10), +15', A12-A27-A1 (57') (Figure 10.5)	153	11530	1238	34750	2/6	Yes
delay between Rijnsweerd and Eemnes (A27), +15', A12-A27-A28-A1 (Figure 10.6) (68')	146	10710	1546	40920	1/6	No*
delay between Gouwe and Oudenrijn (A12), +15', original route (82')	92	12740	1530	41460	1/6	Yes
Average	96	9403	1268	33958	4/13	

* The routes both algorithms found were different, although the travel times were the same.

Also testing along other routes was carried out. In Appendix C the results of these testing procedures are listed. In Table 10.2 a summary of these results is given. Of each route along which has been tested the averages of the number of travel time estimates done by the expert system and graph algorithm are given, together with an average of the total computation time of both methods. Also some kind of indication is given about which alternative route of the expert system was the best route. In Table 10.1 this is indicated as 1/6, which means that 6 alternative routes were generated and the first one of these was the fastest one. The

indication that is given in Table 10.2 is simply the sum of the total routes found (right number) and the sum of the numbers that indicate when the route was found (left number). A value of 4/13 indicates that overall measured the fourth alternative route was the right one, given thirteen routes.

Table 10.2: Average values of testing procedures.

	expert syst. trav.time est.	expert syst. comp. time	graph alg. trav.time est.	graph alg. comp.time	order found	Same route
Muiden-Amerongen	96	9403	1268	33958	4/13	5/6*
Amerongen-Delft	142	10511	1570	46656	8/15	7/7
Amsterdam-Apeldoorn	203	8457	1436	36744	13/42	7/7
Deventer-Gouda	130	10011	1015	32034	12/29	7/8*
Weesp-Moordrecht	466	14664	1302	33238	21/65	9/10*
Total average	**229**	**10899**	**1310**	**36216**		

* The routes were different, although the travel time was the same.

10.5 Extended Dijkstra algorithm versus expert system

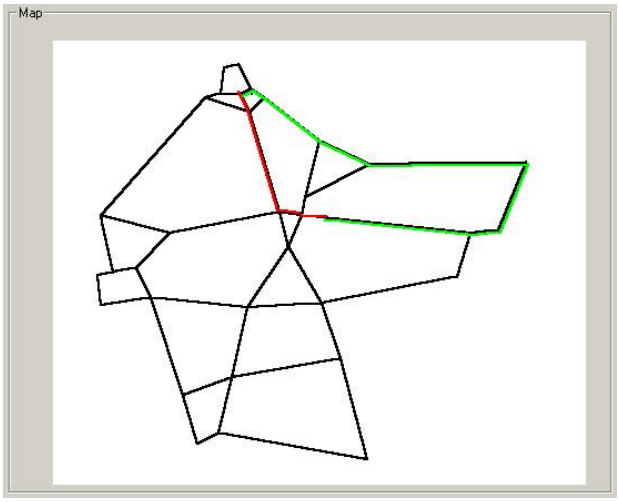
In section 10.2 the criteria on which both methods should be evaluated were given. In this section the results with respect to these criteria will be discussed.

Travel time estimates
The first criterion was the number of travel time estimates, since it is expected the estimation of the travel time will take (by far) the most computation time in a real-time estimation. The data the detection loops generate has to be combined with historical data using some kind of prediction algorithm, which will take a lot of computation time (see chapter 8). When reviewing Table 10.2 it can be seen that the number of travel time estimates is significantly less in the expert system approach. The difference is almost a factor 6. It can also be noticed that in the case of congestion the number of travel time estimates of the expert system only increases if other trajectories along the original route get congested, while the number of travel time estimates of the extended Dijkstra algorithm increases strongly when the total travel time becomes longer, since more possibilities have to be tried.

Overall computation time
The overall computation time (the second criterion that was mentioned in section 10.2) also shows a big difference. As could be expected since the expert system requires less travel time estimates, the expert system needs significantly less computation time. Although dummy data were used for travel time estimation, which resulted in only a little computation time per travel time estimate, the increase in overall computation time due to more travel time estimates can be noticed very clearly. Not only requires the expert system about a factor 3.5 less computation time then the graph approach, the increase in computation time of both methods when more travel time estimates are needed is very obvious (Table 10.2 and appendix C).

(Quality of) shortest routes found
As was stated in section 10.2 it is mathematically proven that the extended Dijkstra algorithm will return the shortest route [ford62]. Consequently it should be investigated if the expert system finds the same (shortest) route. In Table 10.2 and in the tables in Appendix C it can be seen that the few times the routes were different (three times out of approx. 60 routes), the travel times were the same. Since different algorithms were used to find the shortest route both approaches returned a different one, although the other alternative routes

were also returned. Subsequently it can be stated that the quality of the routes found by the expert system is very good.
On the other hand it should also be remarked that the testing procedure influenced the results a little bit. During the testing procedure no scenario's were tried to frustrate the expert system. It would be possible to create such congestion along all reasonable alternative routes that very strange alternative routes would become the best one: in the case of such very unusual traffic conditions the expert system may not find the best route, since no rule has been stated for that situation. When for example travelling from Amsterdam to Bunnik one could create severe congestion along all 'normal' alternative roads such that one would have to travel via Apeldoorn and Arnhem, back to Utrecht to have the fastest route. This route is shown in Figure 10.7. Of course such situations are very rare in reality.
A way to solve the problem that the expert system does not find the best route in case of very severe congestion along all reasonable alternative routes could be to let the expert system find routes first. If no satisfying route has been found by the expert system, the graph algorithm could be applied too (for example, if the percentage of travel time of the original route has dropped below 60% of the newly found route). This way a guarantee can be made about the quality of the route and only in the case of severe congestion along almost all highways the computation time will increase, since both algorithms have to be applied. Also the incremental requirement is still met: first the expert systems returns a best route, which might be improved by the graph algorithm.
Finally, it should also be noticed that the expert system was only tested along routes for which rules were stated. Since the development of the rule base took very much time not for all the trajectories the rules were stated.



Figures 10.7: In Figure 10.7 the shortest route (green route) is shown in case of the (severe) congestion.

Order of found routes (expert system)
With respect to the order in which the expert system generates alternative routes it can be remarked the results are quite well. Most of the time one of the first routes that is generated actually is the fastest route, which would make the incremental requirement feasible. On the other hand it can be noticed that sometimes the best route is one of the last routes found (Table 10.2 and appendix C). For example in the route between Weesp and Moordrecht the best alternative route that is found when the first congestion is created is the eleventh alternative route out of twelve. Although the first alternative route is only one minute slower this performance is not as good as was hoped. Several remarks can be made with respect to this performance. Firstly it should be noticed that when more then one trajectory is congested along a route more file labels will instantiate different rules (see section 6.6). The

order in which these rules fire cannot be regulated in a way the rule with the ‘best’ alternative route fires first, since more then one trajectory is congested and it can not be stated beforehand which alternative route will be most promising in that case. Consequently the alternative routes that are fired by one rule might all be tried first after which the second rule fires which contains the best alternative route.
Secondly, the best alternative route depends on the status of the total road network. Since different road network states imply different best routes and the status of the road network is not know beforehand (when stating the rules) the order of the alternative routes in the rules might not be the correct one in every case. Consequently it might be possible the best alternative route is not the first one found.
The last remark that can be made is concerned with the implementation. In section 6.6 it was stated the travel times of each alternative route should be computed to prevent the travel times of alternative routes being computed that will not make a chance since their detour time is larger then the delay due to the congestion. Since the construction of the rule base took much more time then expected this implementation was not made. Subsequently sometimes alternative routes were tried that should not be tried at all: the performance can be improved by adding this test to all rules.

Chapter 11: Conclusions

In this chapter the results of this research will be discussed and recommendations will be done for future work. In section 11.1 an overview is given of what has been accomplished during this graduation. In section 11.2 recommendations for further research will be given.

11.1 Results

During this graduation project research was done into a dynamic multi modal route planner. A design has been made and a prototype has been implemented using two strategies to find dynamic car routes: 1) using an expert system and 2) using the extended Dijkstra algorithm. Finally, a comparison was made between both algorithms. Considering the goals that were stated in chapter 1, it can be concluded they were all met. In this section the main achievements will be discussed, while in the next section recommendations will be done for further research.

Dynamic multi modal route planner
As far as known no dynamic multi modal route planner was constructed yet (see chapter 2). Consequently KRIS is the first route planner that incorporates dynamic data *and* different modalities. With respect to the dynamic data a new method was identified to collect these data independently. In stead of using dynamic data provided by the traffic information centres (TIC's) and the railway company, travellers can be monitored along their route. Comparing their progress with the planned time schedule of their route, congestion along the highways and train and other public transport delays can be identified. According to R. Pieper [piep01] it is technically feasible to locate the position of a traveller with enough precision for such an application.

Artificial Intelligence approach using expert systems
A new approach has been implemented to perform the actual route planning. Instead of traditional brute force algorithms the knowledge of experts was used to perform the route planning. Secondly, a comparison has been made between this approach and such brute force algorithms.
With respect to the comparison between the expert system and the graph algorithm it can be stated that the expert system performed even better than expected. Although the construction of the rules appeared to take far much time then expected and as a consequence was not completed for every road in the highway network, the performance was very good. The expert system outperformed the shortest path algorithm by far and especially the incremental searching property of the algorithm may prove to be very useful in a real time application. For example when a user is approaching a junction and needs to know quickly which route to take.

Dynamic version of Dijkstra's algorithm
Although very similar to Dijkstra's algorithm, a (brute force) algorithm was constructed that performs dynamic routing. While Dijkstra's algorithm searches the best route in a static two dimensional graph, this algorithm searches the best route in a dynamic three dimensional graph, possibly consisting of the networks of different modalities (see chapter 5). The construction of this algorithm was necessary to be able to compare the expert system approach with a traditional brute force algorithm.

Modular design
A design has been made that makes it possible to use different data sources. Consequently, the functionality of KRIS does not depend on one specific data sources. Different route planners and different information sources that supply dynamic data can be used.

Automatic information retrieval from websites
In an intelligent way data are retrieved automatically from external websites. According to the previously mentioned modular design different websites can be used. Intelligent code was written to extract the needed information from the website that provides public transport routes and similar code was written to retrieve the information from different car route planners.

11.2 Recommendations

Although in the previous section it was concluded that all goals that were stated in chapter 1 were achieved, several issues can be addressed that need further research or can be improved if a real application of the prototype would be developed. Firstly, some implementation issues that can be improved or added are mentioned, followed by recommendations on further research.
First of all, the rule base of the expert system should be expanded to cover the complete network that is monitored by the MONICA system (Figure 6.1). Another quite simple extension would be the construction of files containing the ramps and travel times between them for the highways that are not in the MONICA network. Using these files also routes that start or finish outside the MONICA network can be optimised for the part they travel along this network. An example of such a file can be found in Appendix B. Further improvements in performance can be reached by using local route planners in stead of websites and by incorporating parallel computation of the P+R alternative routes. Finally, as was already mentioned in section 9.5.2, the robustness should be improved. At the moment the route advice changes immediately if the smallest difference in travel time is noticed. A more stable behaviour would be favourable, since congestion is a continuous process. Beside these implementation issues further research should be done to solve the following issues.
It is emphasised once more that the most promising application of KRIS is in a real-time environment, where travellers can consult KRIS through a PITA or other device. KRIS can be a good starting point in the development of such an application, since it has the framework that is needed for such an application. Probably most problems will arise when gathering all needed real time data, e.g. the train delays, MONICA data and other public transport traffic data. These data are expensive from a commercial point of view and it is to be questioned if the data owners will supply these data to a successful system they do not control. Since the possibilities of KRIS are only used to its fullest extent when all these data are incorporated, the most important task is to make such a system independent of these information providers. Detailed research should be done into the possibilities of inferring dynamic data from the user positions (see section 3.8). Using these user positions an independent system can be developed that provides itself with the needed dynamic data.
To be able to use these data to its fullest extent further research should be done into a good travel time predictor. The development of a travel time predictor has been kept out of the scope of this research on purpose, although the travel time prediction module can be seen as the Achilles' heel of KRIS: without good travel time predictions the shortest route in time can not be found. Travel time prediction is mainly a civil engineering discipline and as was stated in chapter 8 within TRAIL much research is done in this field. Of course meanwhile no good predictor is available less accurate travel time estimates can be used. For example the kilometres of congestion that are supplied by the Traffic Information Centre can be translated into travel times.

Also further research should be carried out into the module that performs the dynamic routing for the public transport, to handle public transport delays. As we all know this is a very actual issue in the Netherlands, where about 20% of the trains is delayed. About 13% of all connections is missed, while for some trajectories this percentage is between the 40 and 50% [rov00]. In the design that was given in chapter 5 a module that takes into account these delays has been incorporated, but no design of such a module made nor implemented yet.
Another interesting extension of KRIS would be to develop a module that assigns probabilities to the different routes it advises. For example, the probability of congestion along each car trajectory can be given and for each public transport leg the probability of delays and the probability of missing a connection can be given. Using these probabilities the traveller can decide better which modality and route to take. When he has an important appointment he might want to travel along the most ‘secure’ route, while someone who has no important date would like to travel along the fastest route, although the probability of severe congestion is present.

Bibliography

[boul92] L. Boullart, A. Krijgsman and R.A. Vingerhoeds (1992), *Application of artificial intelligence in process control*, Pergamon Press, Oxford, England.

[bovy00] P.H.L. Bovy, R. Thijs (2000), *Estimators of travel time for road networks, new developments, evaluation results, and applications*, Delft University Press, Delft, the Netherlands.

[clips] G. Riley, *CLIPS*, http://www.ghg.net/clips/CLIPS.html

[good98] M.T. Goodrich, R. Tamassia (1998), *Data structures and algorithms in java*, John Wiley & Sons, New York, USA.

[good99] M. T. Goodrich, M. Handy, B. Hudson, and R. Tamassia, *Accessing the Internal Organization of Data Structures in the JDSL Library.* In *Proceedings of ALENEX conference on January 15th and 16th, 1999*. Baltimore, Mariland. Also http://www.cs.brown.edu/cgc/jdsl/pub.html

[grol97] H.J.M. Grol (1997), *On-line network state estimation and short-term prediction*, DACCORD Deliverable D05.2, Telematics Application Programme project TR1017, Brussels.

[king90] J. H. Kingston (1990), *Algorithms and data structures: design, correctness, analysis'*, Addison-Wesley, University of Sydney, Australia.

[kots97] A. Kotsiales, C. Diakaki, M. Papagerorgiou (1997), *The METANET Traffic Model*, Annex A of Co-ordinated Control Strategies, Deliverable D06.1, Telematics Application Programme project TR1017, Brussels.

[kroon99] R. Kroon, J.M.P. van Waveren (1999), *Reisbegeleidingssysteem documentatie*, TU Delft, faculty Computer Science, the Netherlands.

[lint00] J.W.C. van Lint, *Robust and adaptive travel time prediction with neural networks. In Proceedings of TRAIL 6th annual congress, December 12th 2000.* The Hague/Scheveningen, The Neterlands.

[oj99] T. Ojala, A. Lumiaho (1999), *Personal mobile traveller and traffic information service. Final Report.*, Nokia Mobile Phones, Finland. See also http://www.promise.cellulardata.com

[park00] (2000), *PARKIR, report on parking possibilties near trainstations*, OVR.

[piep01] R. Pieper (2001), *Bewust op weg, report for the Dutch Ministry of Public Works*, Bloemendaal, the Netherlands.

[pop00] E. Poppe (2000), *De weg naar de ideale routeplanner op cd-ROM en Internet*, University of Utrecht, graduation thesis.

[reijm99] J. J. Reijmers (1999), *Verkeersbegeleidingssystemen*, collegedictaat TU Delft.

[rog99] S. Rogers, C. Fiechter and P. Langley (1999), *A route advice agent that models driver preferences*, DaimlerChrysler Research and Technology Center, Palo Alto, USA. Also http://pc19.rtna.daimlerchrysler.com/~rogers/ss-99.

[rov00] D. v.d. Spek e.a. (2000), *Kwaliteitsthermometer NS, onderzoek punctualiteit treinen en kwaliteit van de informatievoorziening, rapportage winter 1999/2000,* Rover, Amersfoort, Netherlands.

[sab92] M.T. Saborido (1992), *An introduction to expert system development*. In *Application of Artificial Intelligence in process control*, L. Boullart, A. Krijgsman en R.A. Vingerhoeds, Pergamon Press, Oxford, UK.

[sto00] H. Stoelhorst (2000), *De markt wordt aangeslingerd.* In *Wegen*, april 2000, pages 12-15.

[trail99] *Conference proceedings of the 5th annual congress, December 1999*, TRAIL Research School.

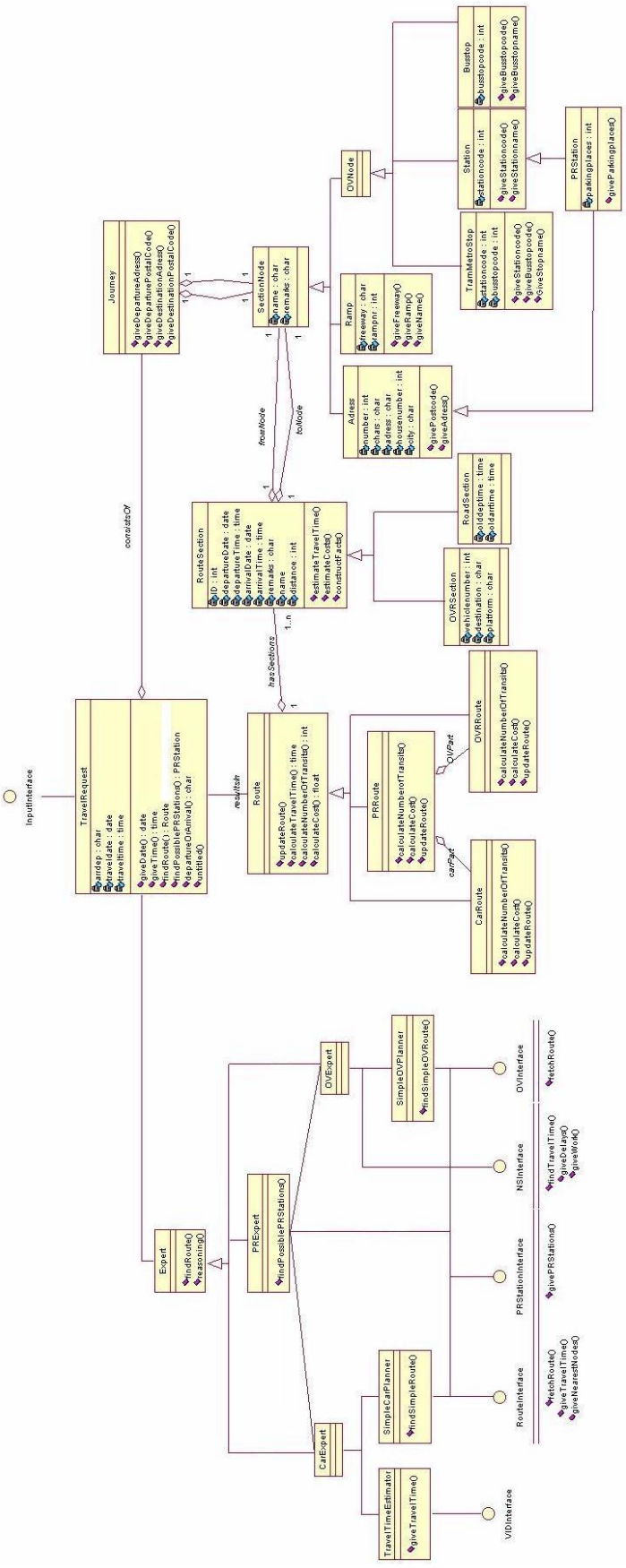
[trav] Travelstar: http://www.travelstar.nl/index.htm.

[yil00] 'DTGS, een ontwerp voor een dynamisch reisbegeleidingssysteem', Erdal Yildiz, afstudeerverslag TH Rijswijk / TU Delft, 14 oktober 2000.

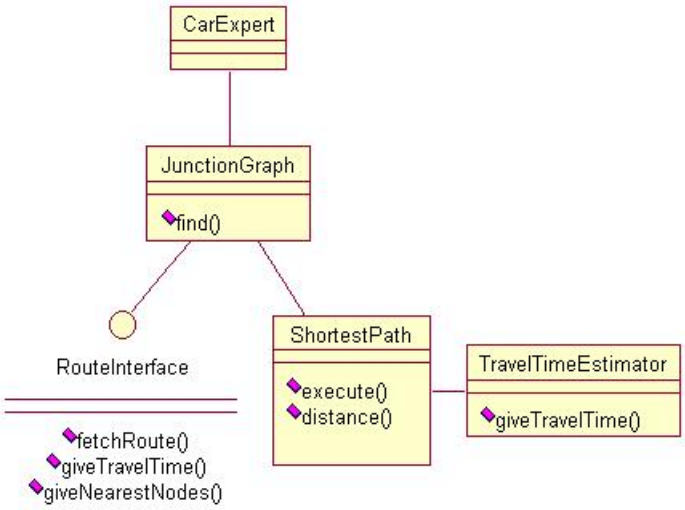
[vark00] 'Personal and dynamic travel information, client aspects', R.J. van Vark M.Sc., EDS/T&T Solution Utrecht, TRAIL Research School, Delft/Rotterdam,Groningen, 30 september 2000.

[vid01] Traffic Information Service of the Netherlands, www.vid.nl.

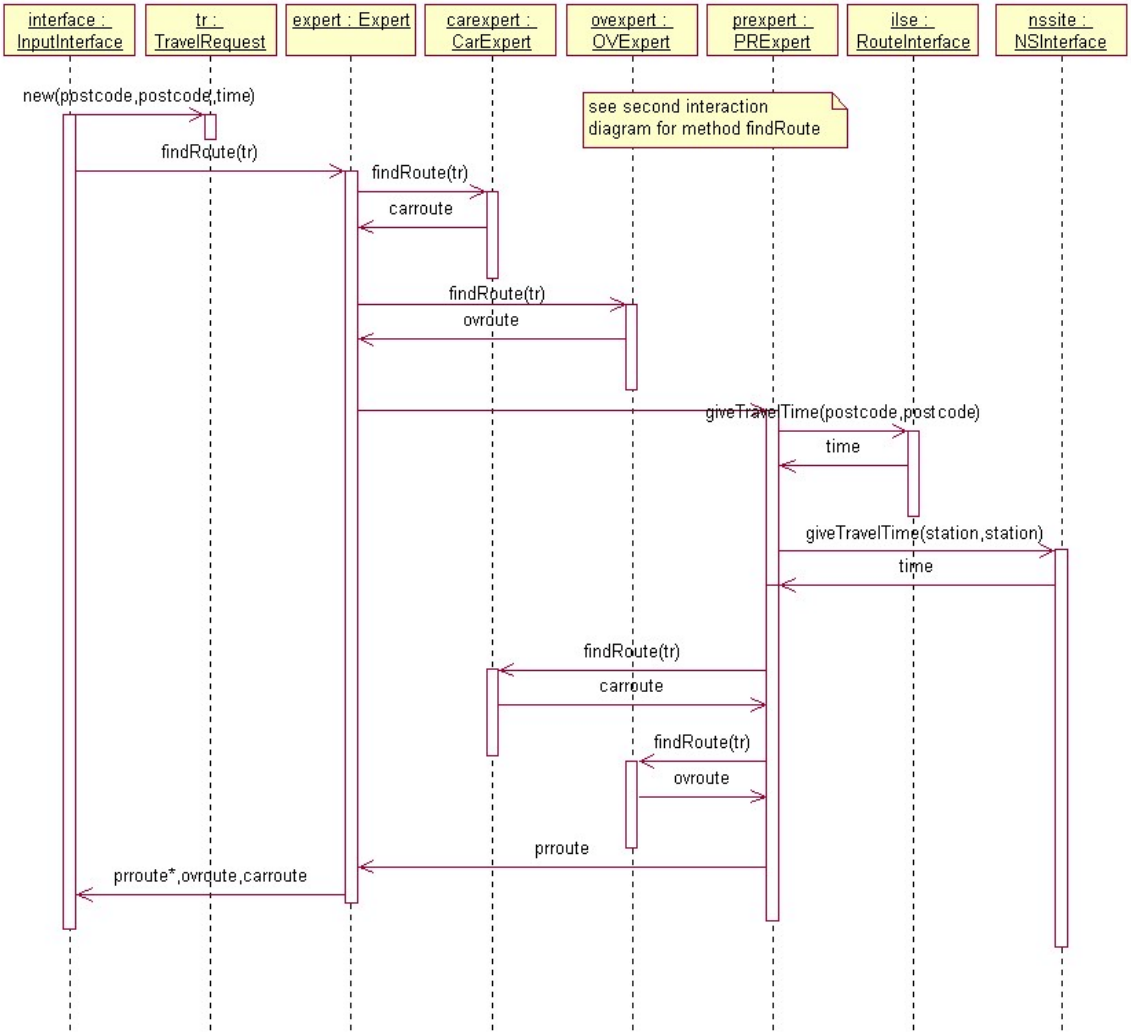
Appendix A: Object model



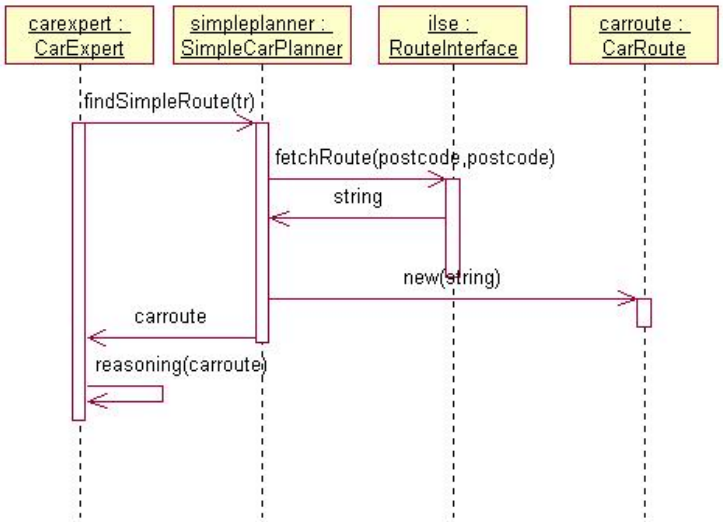
A.1. Class diagram.



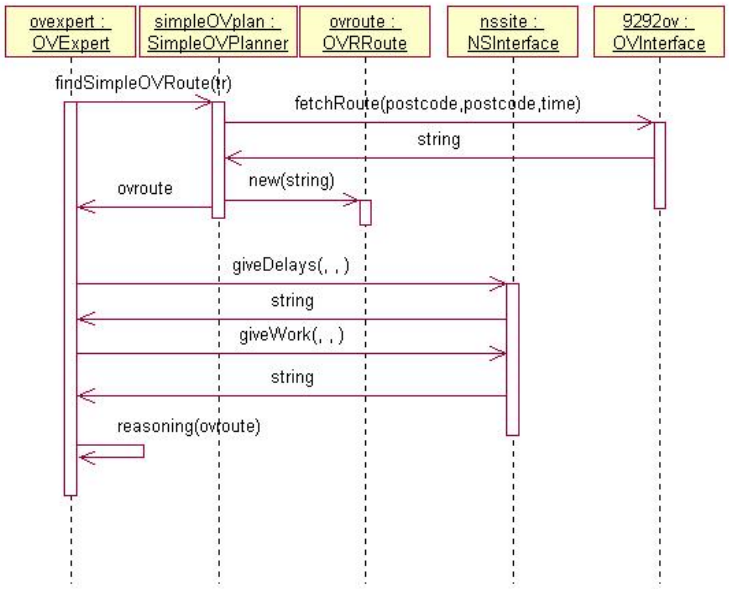
A.2: Class diagram of extended Dijkstra implementation.



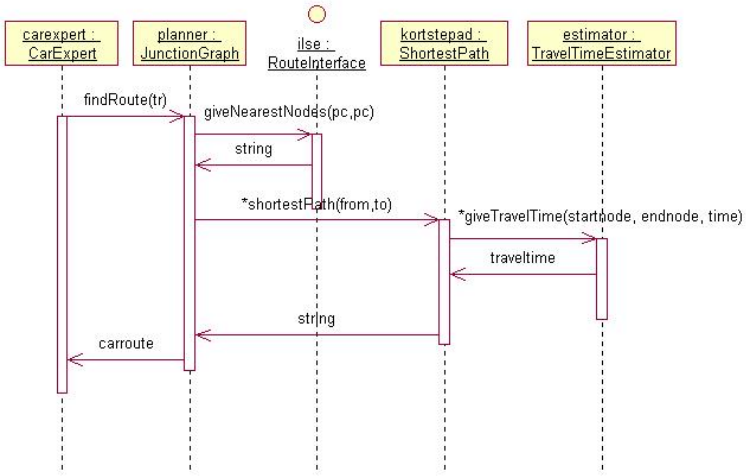
A.3: Main interaction diagram.



A.4: Interaction diagram for finding car route.



A.5: Interaction diagram for finding OV route.



A.6: Interaction diagram for finding car route using the extended Dijkstra algorithm.

Appendix B: Travel time file

A4

knooppunt nieuwe_meer ()				
sloten (1)	1	2	1	3
knooppunt badhoevedorp ()	1	1	1	3
schiphol (2)	3	3	3	5
hoofddorp (3)	2	2	2	4
nieuw-vennep (4)	3	5	3	7
burgerveen ()	1	2	1	4
roelofarendsveen (5)	3	5	3	6
hoogmade (6a)	5	1	5	1
zoeterwoude_rijndijk (6b)	3	5	3	5
zoeterwoude_dorp (7)	1	2	1	2
leidschendam (8)	5	6	5	6
knooppunt prins_claus ()	1	1	1	3
knooppunt ypenburg ()	1	2	2	3
rijswijk_centrum (9)	0	2	2	2
plaspoelpolder (10)	1	1	1	1
rijswijk (11)	1	1	1	1

____

rijswijk (11)				
plaspoelpolder (10)	1	2	1	1
rijswijk_centrum (9)	1	2	1	2
knooppunt ypenburg ()	0	1	0	1
knooppunt prins_claus ()	1	3	1	3
leidschendam (8)	1	3	1	3
zoeterwoude_dorp (7)	5	10	5	7
zoeterwoude_rijndijk (6b)	1	3	1	2
hoogmade (6a)	3	6	3	4
roelofarendsveen (5)	3	6	3	5
burgerveen ()	5	10	5	7
nieuw-vennep (4)	1	3	1	2
hoofddorp (3)	3	7	3	4
schiphol (2)	2	5	2	5
knooppunt badhoevedorp ()	3	6	3	6
sloten (1)	1	3	1	3
knooppunt nieuwe_meer ()	1	3	1	3

Appendix C: Results

Table C.1: Results of tests on route Amerongen – Delft, at 16.03.2000, departure 18.10.

	exp.syst. trav. est.	exp.syst. comp.time	graph alg. trav. est.	graph alg. comp. time	order found	routes are the same
original route	29	8400	329	18730	0/0	Yes
delay Gouwe-Prins Claus (A12), +15' (74')	169	10540	1720	48230	1/5	Yes
delay Gouwe-Terbregse-plein (A20), +15' (77')	172	10490	1753	43720	4/5	Yes
delay Ridderkerk-Terbreg-seplein (A16), +15' (78')	167	10760	1844	46080	5/5	Yes
delay Gorinchem-Ridder-kerk (A15), +15' (83')	152	10820	1804	52350	1/5	Yes
delay Oudenrijn-Gouwe (A12), +15' (93')	153	11200	1778	47560	2/5	Yes
delay Holendrecht-Badhoe-vedorp (A9), +15'(98')	153	11370	1764	69920	3/5	yes
Average	142	10511	1570	46656	8/15	

Table C.2: Results of tests on route Amsterdam-Apeldoorn, at 22.04.2000, departure 10.10.

	exp.syst. trav. est.	exp.syst. comp.time	graph alg. trav. est.	graph alg. comp. time	order found	routes are the same
original route (56')	14	5550	213	12580	0/0	Yes
delay Diemen-Eemnes (A1), +15' (72')	266	8350	1565	37730	3/7	Yes
delay Rijnsweerd-Hoevela-ken (A28), +15' (74')	256	8840	1583	39050	4/7	Yes
delay Rijnsweerd-Eemnes (A27), +15' (80')	217	7850	1593	40260	0/7*	Yes
delay Watergraafsmeer-Diemen (A1), +15' (84')	240	8240	1676	41790	1/7	Yes
delay Holendrecht-Water-graafsmeer (A9), +15' (89')	238	8400	1711	42900	6/7	Yes
delay Oudenrijn-Lunetten (A12), +15' (95')	191	11970	1711	42900	0/7*	yes
Average	203	8457	1436	36744	13/42	

* No one of the routes was better, so the original route that was found first was returned.

Table C.3: Results of tests on route Deventer-Gouda, at 22.04.2000, departure 10.10.

	exp.syst. trav. est.	exp.syst. comp.time	graph alg. trav. est.	graph alg. comp. time	order found	routes are the same
original route (76')	0	6810	186	13400	0/0	Yes
delay Hoevelaken-Rijns-weerd (A28), +15' (79')	153	9990	1058	30490	1/4	Yes
delay Eemnes-Rijnsweerd (A27), +15' (83')	146	10110	1075	31530	2/4	Yes
delay Beekbergen-Wat-erberg (A50), +15' (91')	114	10220	1090	30480	0/4*	Yes
delay Hoevelaken-Rijns-weerd) +10' (95')	158	10600	1170	33280	3/4	Yes
delay Holendrecht-Diemen (A9), +15' (96')	143	10220	1141	37790	1/4	Yes
delay Lunetten-Oudenrijn (A12), +15' (97')	175	11150	1175	43000	5/5	yes

delay Amstel-Watergraafs-meer (A10), +15' (104', 109')	154	10990	1227	36300	0/4*	No**
Average	130	10011	1015	32034	12/29	

* No one of the routes was better, so the original route that was found first was returned.
** The routes that were returned were different. See figures.

Table C.4: Results of tests on route Weesp-Moordrecht, at 22.04.2000, departure 10.10.

	exp.syst. trav. est.	exp.syst. comp.time	graph alg. trav. est.	graph alg. comp. time	order found	routes are the same
original route (45') A9-A2-A12-A20	0	5880	216	13230	0/0	Yes
delay Holendrecht-Ouden-rijn (A2), +20' (57') A9-A2-A10-A4-A12-A20	448	14120	1240	29770	11/12*	Yes
delay Amstel-Nieuwe Meer (A10), +15' (58') A9-A1-A27-A12-A20	446	14450	1269	30860	1/12	Yes
delay Eemnes-Rijnsweerd (A27), +15' (59') A9-A4-A12-A20	420	13740	1113	26700	9/12	Yes
delay Holendrecht-Bad-hoevedorp (A9), +15' (67'), original route	386	14230	1199	30150	0/12	Yes
delay Oudenrijn-Gouwe (A12), +20' (73'), A9-A10-A4-A12-A20	610	16370	1443	35530	15/16	Yes
delay Prins Claus-Gouwe (A12), +15' (79'), A9-A10-A4-A13-A20	592	16540	1488	37340	2/18	yes
delay Badhoevedorp-Prins Claus (A4), +20' (87') original route	553	16590	1572	43010	0/16**	yes
delay Oudenrijn-Gouwe (A12), +10' (91'), A9-A2-A27-A15-A16-A20	601	17530	1736	43280	2/16**	Yes
delay Oudenrijn-Everdin-gen (A2), +15', A2-A12-A27-A15-A16-A20 and A1-A28-A27-A15-A16-A20	599	17190	1743	42510	3/16**	No***
Average	466	14664	1302	33238	21/65	

* This alternative route was 1 minute faster then 1/12
** The delay along the A12 between Prins Claus and Gouwe was removed after the 2/18 alternative route was found. Consequently less alternative routes are generated, since no rules for this delay are initiated.
*** different alternative routes have the same travel time (92').

Appendix D: Paper

Intelligent dynamic route planning

G. Eggenkamp L.J.M. Rothkrantz

Delft University of Technology, Zuidplantsoen 4, 2628BZ Delft, the Netherlands

Abstract

In this paper the possibilities of artificial intelligence and especially of expert systems in the field of route planning using dynamic traffic data are explored. An expert system that has been built to perform dynamic routing and a dynamic route planner using a (traditional) shortest path algorithm are introduced. Using both implementations a comparison is made between the expert system approach and the shortest path approach. It is concluded that the expert system shows great potential. It outperforms the shortest path algorithm in computation time and the routes the expert system finds are indeed the shortest routes.

1. Introduction

Currently no dynamic route planners are available. Although the highway network in the Netherlands is flooded with cars and is subject to heavy congestion in both rush hours no planner is available that finds the shortest route in this (congested) network. The only option some route planners and car navigation systems offer is to 'block' roads that are congested and to find the best alternative route without using this road. Consequently a route is advised that might well take longer then the route along the congested road, since this road is not even considered anymore. Since dynamic data are available from the MONICA monitoring system (the detection loops under the highways) a study has been carried out to develop such a dynamic route planner.
When research was carried out to the performance of shortest path algorithms, like Dijkstra's algorithm, to find the shortest route while using dynamic data, it showed that the computation time degrades significantly when dynamic data are incorporated. Consequently, other possibilities were investigated and since humans are quite well capable of finding alternative routes in the case of congestion it was decided to study the feasibility of an artificial intelligence approach. In this paper the feasibility of an expert system in a dynamic route planner is discussed and a comparison is made with a 'regular' shortest path algorithm.
This paper will start with a problem definition (section 2) and an introduction of a shortest path algorithm that can be used to find the shortest path in a dynamic network (section 3). In section 4 an introduction of the expert system that was constructed is given, while in section 5 the results of both approaches are presented. In section 6 some conclusions are given.

2. Problem definition

The highway network can be represented by a graph, as with static routing problems. The highway network that was considered in the research is the network that is being monitored by the MONICA system, since only dynamic data are available of these roads. In Figure 1 this network is shown. Somehow the dynamic aspect of the data has to be taken into account. The travel times between different edges (cities, junctions) change in time, and these changes have to be taken into account and incorporated in the graph. When, for example, travelling from Amsterdam to Delft in the morning rush hour, a

departure of only 5 minutes later, can affect the travel time by more then 20 minutes, since major congestion may have occurred along the route during these 5 minutes. For example an accident might have happened or a sudden peak in cars that want to access the highway may have occurred.
A space time extended network (STEN) explicitly represents time by having a complete layer of all nodes of the physical network per time period. The first occurrence of STEN in literature can be found in [4]. Other applications of space-time expanded networks can be found in [1,3,6]. In all these publications time expanded networks were used to solve traffic assignment models, which are dynamic flow problems. In dynamic route planning only the shortest path has to be found, no dynamic flow problem has to be solved. Consequently, the same approach can be used, only with a flow of 1 for all links.
Concordant to these publications the space-time expanded network can be constructed as follows:

- for each period p create a complete layer of all nodes of the physical network,
- for each node in period p (all nodes with the same t), create links to the nodes it is connected with in the physical network in the corresponding period 'layer' p + d, with d the travel time when starting at period p,
- for each node in period p create a link to the same node in period p + 1 (it is also possible to stop in a node).

An example of a network constructed this way is shown in Figure 2. The original graph consisting of nodes A to G is repeated for each time interval. The edges between the nodes A to G (the lowest graph at t = 10:01) represent which nodes are connected to each other. For clarity these edges have been kept in the different layers of the graph to show these connections. The thicker links that intersect the different layers are the actual road connections. Their length (and thus the layer to which they go) represents the travel time when starting at the time of the layer in which they start. For each layer for all nodes all outgoing links are constructed and labelled according to the travel time at that moment. It should be noticed that in Figure 2 not all the links are shown, since that would have resulted in a cluttered figure.

3. The extended Dijkstra algorithm

The most secure way to find an optimal route from an origin to a destination at a specific time of the day would be to find the optimal route in the graph that was constructed in the previous section. In [4] a proof is given that a dynamic routing problem that is expanded in the way described can be solved using static shortest path algorithms.

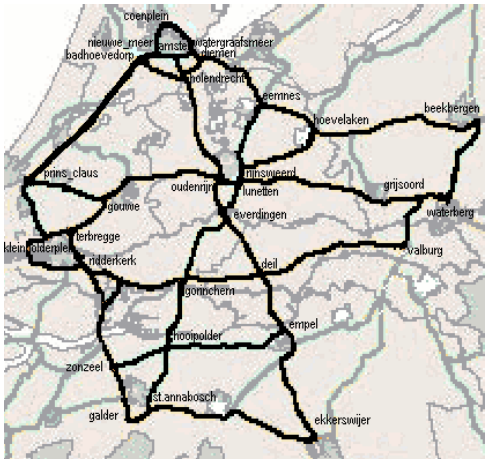


Figure 1: The freeway network that is monitored by the MONICA system.

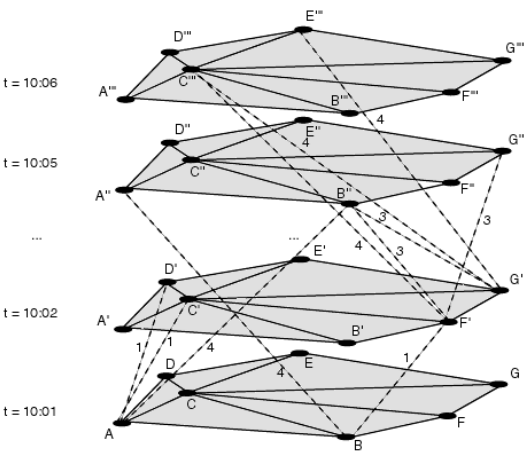


Figure 2: A space-time expanded network.

In practice the graph that is proposed in section 2 better can not be constructed, since this would require a lot of computation time. It would be far more efficient if travel times only were estimated if they are really needed. Consequently an algorithm was constructed that finds the shortest path in this 3-dimensional graph and which only estimates travel times if necessary. When the algorithm was constructed and was reviewed thoroughly, it was discovered it differs with Dijkstra's algorithm only slightly. Consequently, it was called the 'extended Dijkstra algorithm'. Since the Dijkstra algorithm is widely known, no explanation is given here. It can be found in [3].
It should be noticed that no research was done into the estimation of travel times, which is a very complex process. During this project we focused on one main aspect: route planning. Consequently, it was assumed travel time estimates were available and a set of historical data was used as 'dummy' data.

4. Expert system

In this section the expert system that has been constructed is introduced. As was stated in section 2 the problem domain of the expert system is given by the road network that is given in Figure 1. The expert system should find the fastest route in this network.

4.1 Knowledge elicitation

The knowledge the expert system should possess consists of alternative routes in the case of congestion along a part of a road. This knowledge can be made explicit in two ways. Firstly, experts can be interviewed. These experts should be experienced 'traffic jam travelers' that often have tried alternative routes in the case of congestion along a part of the freeway they normally use. Secondly, the map of the Netherlands combined with historical traffic data can be investigated, to see which alternatives are reasonable in the case of a traffic jam. In this project, the second approach was chosen. The reason to do this was as follows. Travelers are not able to monitor the routes they did not choose. When they have chosen an alternative, afterwards they do not know if it was faster then the original route or other alternatives (unless they know another traveler, who tried the alternative at the same time). Consequently, the perception the traveler has of the quality of alternatives he tried can be wrong, since it may also be influenced by other incentives then the shortest travel time.

4.2 Level of detail

For the level of detail in which congestion in the road network is monitored route sections between junctions where one can change freeways were chosen. Since routes are only optimised in the freeway network (and not considering secondary roads), only at junctions the route can be changed. Since secondary roads are not taken into account, it is not interesting to construct rules on the basis of congestion between two ramps: for all ramps that are between two junctions, the same rules would be constructed, since only at the first junction after the ramp it is possible to change the route. In Figure 1 the different road parts between junctions can be found (the junctions are identified by their names).

4.3 Route representation

The expert system is provided with a number of route parts (trajectories), that each have a label, which can be 'route', 'file', 'entrance' or 'exit'. These route parts are delimited by two junctions. An example of such a route can be found in Figure 3. The route on the map was generated by a static route planner and was translated to a route for the expert system. Right of the figure the translation of the route can be found.

(entrance coenplein nieuwe_meer)
(route nieuwe_meer badhoevedorp)
(route badhoevedorp burgerveen)
(route burgerveen prins_claus)
(route prins_claus ypenburg)
(exit ypenburg kleinpolderplein)

Figure 3: The route from Amsterdam to Delft and the format that is given to the expert system.

It was chosen to keep the choice whether there is congestion along a trajectory, outside the domain of the expert system. A separate module was constructed in which it is decided if the delay is significant enough to consider the trajectory congested. This module provides the expert system the route with its appropriate labels.

4.4 Construction of the rule base

For each trajectory along the road network rules were made stating which alternative to take if the trajectory was congested. Since there are 92 edges in the network that is monitored by the MONICA system (Figure 2) the construction of these rules was a time consuming task. The alternative route that can be taken when a route part is congested depends on the direction one is coming from and the direction one is going to: the rules for alternative roads depend on the previous and following route parts. The different steps, which are needed to construct the different rules for each trajectory are given in the following action list.

1. Determine the possible directions where one can be coming from,
2. determine the possible directions where one can be going to,
3. determine the different alternative routes for each possible route, by investigating the map and historical data,
4. calculate the ‘detour time’ of each alternative: the time needed to make the detour in the best case (no congestion along the alternative route),
5. order the different routes according to this ‘detour-time’,
6. when the ‘detour-time’ is too large, do not use the alternative,
7. check with reports of car drivers if no routes are missing.

The first two steps are very straightforward, the possible directions can be found by having a look at the map. The third step requires by far the most time: in this step the different alternative routes have to be chosen. When these routes are known, their travel times can be computed. The fifth step is important to find the best route as quickly as possible. This property can be very useful if the dynamic route planner is used in a real-time environment and there is a time constraint. When alternatives become available very fast, while searching is continued for better alternatives, the best route found so far can be used if the time constraint has to be met. Of course, it would be optimal if the first route found also is the best route and this is examined using this parameter. Consequently, the order in which the alternatives are searched should depend on the travel times and the chance of congestion along these alternatives, when there is congestion along the trajectory for which alternatives are searched.

5. Results

To be able to compare both methods the following four parameters were chosen: 1) the number of travel time estimates, 2) overall computation time, 3) shortest route found and 4) order of found routes.

The number of travel time estimates parameter was chosen since it gives an indication of the performance of the algorithm. Since the travel time estimation is the process that needs the most computational time the number of travel time estimates will strongly indicate the total computational time needed.
The overall computation time parameter is included to be able to judge the performance of the expert system. It could be possible, the expert system approach needs only few travel time estimates, but is very slow itself, since the rule base is very large.
The third variable on which the methods will be compared is the shortest route that is found. Since it is mathematically proven that the extended Dijkstra algorithm will return the shortest route this parameter is only applicable to the expert system.
The last variable which will be carefully examined is the order in which alternative routes are given. This aspect is also only applicable to the expert system, since Dijkstra's algorithm does not return any other route then the best one. It should be examined how many alternatives have to be computed to find the shortest one. As was stated in section 4.4 this can be important in a real-time environment.

5.1 Testing protocol

To test both approaches several departure and destination addresses were chosen between which the shortest route had to be found. Firstly the routes were searched on a free-flow network, without congestion. Congestion was created along one of the trajectories in the shortest route that was found and both algorithms were applied again. Again congestion was created along one of the trajectories of the newly found route and both algorithm were applied. This process was repeated until all trajectories were delayed. In Figures 5 and 6 the first two iterations of this process are illustrated. In Figure 5 the free flow route that was found is shown. Congestion was created between the Prins Claus and Badhoevedorp junctions and both algorithm were applied. The shortest route found now is shown in Figure 6. Now congestion was created between the Holendrecht and Diemen junctions and the same process was repeated. In Table 2 an overview of the results of the first three steps of this testing procedure for the route between Zoetermeer and Muiden is given.

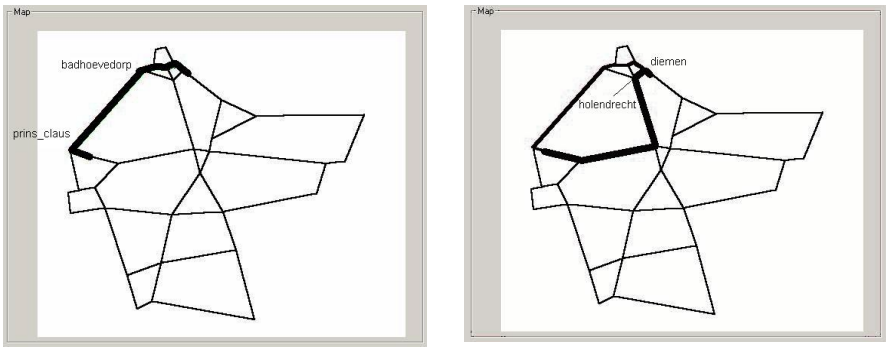


Figure 5 and 6: Free flow route and route if there is congestion between the Prins Claus and Badhoevedorp junctions.

Table 2: Test results of route between Zoetermeer and Muiden.

	exp. syst. trav. est.	exp. syst. comp.time	graph alg. trav. est.	graph alg. comp.time	Order found	routes are the same
Original route, A12-A4-A10-A1 (48')	0	9500	722	17470	0/0	Yes
Delay between prins claus and badhoevedorp (A4), +20',A12-A2-A9-A1 (53')	93	10000	1286	34270	1/3	Yes
Delay between holendrecht and diemen (A9), +15', A12-A2-A10-A1 (54')	94	10490	1285	34880	3/3	Yes

5.2 Results

In Table 3 the results of this testing procedure for different routes are shown. In the most left column the routes are denoted together with the number of iterations with congested trajectories that was carried out. Columns 2 to 5 show the average number of travel time estimations and the average computation time of both methods. In column 6 some kind of indication is given which alternative of the expert system was the best route. In Table 2 this was indicated as x/6 for each route, which means that 6 alternatives were generated and the first one of these was the fastest one. The indication that is given in Table 3 is simply the sum of the total routes found (right number) and the sum of the numbers that indicate when the route was found (left number). A value of 4/13 indicates that overall measured the fourth alternative was the right one, given thirteen routes. The last column indicates the number of correct routes out of the number of total routes found.

Table 3: Average values of testing procedures.

	expert syst. travtime est.	expert syst. comp. time	graph alg. travtime est.	graph alg. comp.time	order found	same routes
Muiden-Amerongen (6)	96	9403	1268	33958	4/13	5/6*
Amerongen-Delft (7)	142	10511	1570	46656	8/15	7/7
Amsterdam-Apeldoorn (7)	203	8457	1436	36744	13/42	7/7
Deventer-Gouda (8)	130	10011	1015	32034	12/29	7/8*
Weesp-Moordrecht (10)	466	14664	1302	33238	21/65	9/10*
Total average	**229**	**10899**	**1310**	**36216**		

* The routes were different, although the travel time was the same.

Table 3 shows the expert system requires significantly less computation time then the shortest path algorithm. The overall computation time is a factor 3.5 less, while the difference of the number of travel time estimates is almost a factor 6. Since it is expected the estimation of the travel time will take (by far) the most computation time in a real-time estimation it can be expected the expert system will perform even better when used together with MONICA data: the data the detection loops generate have to be combined with historical data using some kind of prediction algorithm, which will take a lot more computation time then currently was needed, since dummy data were used.
In Table 3 it can be seen that the few times the routes were different (three times out of approx. 60 routes), the travel times were the same. Since different algorithms were used to find the shortest route both approaches returned a different one, although the other alternatives were also returned. As a result it can be stated that the quality of the routes found by the expert system is very good. On the other hand it should also be remarked that the testing procedure influenced the results a little bit. During the testing procedure no scenario's were tried to frustrate the expert system. It would be possible to create such congestion along all reasonable alternatives that a very strange alternative would become the best one. When for example travelling from Amsterdam to Utrecht one could create severe congestion along all 'normal' alternative roads such that one would have to travel via Apeldoorn and Arnhem, back to Utrecht to have the fastest route. Of course such situations are very rare in reality.
With respect to the order in which the expert system generates alternative routes it can be remarked the results are quite well. Most of the time one of the first routes that is generated actually is the fastest route. On the other hand it can be noticed that sometimes the best route is one of the last routes found. This is a consequence of the unpredictable behaviour of congestion. As was stated in section 4.4 it was tried to rank the different alternative routes in such a way that the alternative with the highest chance of being the best one was tried first. Since a chance guarantees nothing sometimes other routes are better. Especially when more then one trajectory is congested along a route the best alternative can be one of the last ones tried. Two or more trajectories are congested, so two or more file labels will instantiate different rules. The order in which these rules fire cannot be regulated in a way the rule with the 'best' alternative fires first, since more then one trajectory is congested and it can not be stated beforehand which alternative will be

most promising in that case. Consequently the alternative that are fired by one rule might all be tried first after which the second rule fires which contains the best alternative.
The last remark that can be made is concerned with the implementation. In section 4 it was stated the travel times of each alternative should be computed to prevent the travel times of alternatives being computed that will not make a chance since their detour time is larger then the delay due to the congestion. Since the construction of the rule base took much more time then expected this implementation was not made. Subsequently sometimes alternatives were tried that should not be tried at all.

6. Conclusion

In this paper the possibilities of an expert system in the field of dynamic route planning were discussed and a comparison was made between a shortest path algorithm and the expert system. The expert system showed great potential. Not only performs the expert system much better with respect to computation time, the routes the expert system returns are as good as the routes the conventional shortest path algorithm computes and the expert system shows great possibilities when real time constraints are placed. The expert first generates all possible solutions and then computes their travel time one by one. As soon as the travel time of a solution has been computed the solution becomes available.
The most important drawback of the expert system approach is the construction of the rules. This is a very intensive process and requires a lot of time.

References

[1] H.K. Chen. *Dynamic travel choice models: a variational inequality approach*. Springer, Heidelberg, 1999.

[2] G. Eggenkamp. *KRIS: Knowledge based routing information system*. Graduation thesis, TU Delft, 2001.

[3] J.R. Evans and E. Minieka. *Optimization algorithms for networks and graphs.* Marcel Dekker Inc., New York, 2nd edition, 1991.

[4] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, New Yersey, 1962.

[5] B. Ran and D.E. Boyce. *Modeling dynamic transportation networks: an intelligent transportation system oriented approach.* Springer-Verlag, Berlin, 2nd edition, 1996.