Hierarchical Routing System using Ant Based Control

Henrik Dibowski

THESIS

Delft, July 2003

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

Dresden University of Technology, Germany
Faculty of Computer Science

Scientific Coordinator:
Dr. Drs. L.J.M. Rothkrantz

TU Delft
Delft University of Technology

TECHNISCHE UNIVERSITÄT DRESDEN

Keywords: Dynamic routing system, Hierarchical routing, Navigation system, Hierarchical network, Intelligent agents, Ant Based Control, Distributed routing

Acknowledgements

The Socrates/Erasmus exchange programme gave me the opportunity to do this thesis at Delft University of Technology. Therefore I would like to thank all people who were involved in the organisation of my five month stay in the Netherlands. Especially I would like to thank Mr. Dr. Petersohn, my mentor at Dresden University of Technology, and Mr. de Vries, who is responsible for international student affairs at Delft University of Technology.
Specifically I would like to thank my mentor Mr. Dr. Rothkrantz at Delft University of Technology for all his guidance and advice during this project. He provided me the very interesting topic of this thesis and aroused my interest in computational intelligence. I really enjoyed our weekly sessions which were both enjoyable and inspiring. Although Mr. Dr. Rothkrantz is a very busy man, I could always count on him to help. It is really admirable how somebody can put so much energy and interest on a project. I am also very thankful for his commitment to introduce me to other students and staff members what made my time at the university very pleasant.
Furthermore I would like to thank all my friends in Delft that made my stay here to (till now) the best time of my live. It was such an amazing and wonderful experience to be together with people from so many different countries from all over the world. I will always positively remember the marvellous five month in Delft which widely broadened my mind.

Summary

Nowadays traffic jams and slow-moving traffic are a big problem in many countries. Every day during the rush hours many streets are congested what results in big economic and environmental damages. Different approaches are followed to solve the problem. One way is the development of dynamic routing systems which use dynamic information about the traffic conditions on the streets to route vehicle on alternative ways around congestions. By that the road capacity is exploited in a more optimal way and congestions can be reduced.
In this thesis the design of a dynamic routing system, called Hierarchical routing system, has been developed which follows a new approach. It splits traffic networks into several smaller and less complex networks by introducing a hierarchy. The Hierarchical routing system therefore is designed to be a distributed routing system which consists of several parts where each of it is responsible for one network of the hierarchical network.
The route optimization is done with the ABC-algorithm, a decentralized routing algorithm, which uses intelligent agents which explore the network and find the shortest routes in time. This algorithm was derived from the behaviour of ants in nature. For the application in a hierarchical network the ABC-algorithm has been adapted. The concept of virtual nodes and time columns has been added and time agents have been introduced. Routes, which start in one network of the hierarchical network and end in another network, are computed as composed route out of several routes. An algorithm has been developed that computes the composed routes with the cooperation of the distributed parts of the Hierarchical routing system.
For collecting real time traffic information the vehicles themselves are used. They record the route they follow together with the needed time and send it at regular intervals to the Hierarchical routing system. That information is used to compute travel time estimates from intersection to intersection for every street of the network.
The concept of hierarchical routing combined with the ABC-algorithm ensures an excellent scalability of the routing system. By that the system should be capable of routing vehicles in real time even in very complex networks. Besides, the design of the Hierarchical routing system ensures a high robustness against failures of parts of the distributed system.

Table of Contents

Chapter 1 : Introduction

1.1 Problem setting

Nowadays the car traffic is a very big problem in many countries. Because flexibility and by that mobility is a must in our modern market economy many people are dependent on travelling long distances every day to go to their job and return home. Today much more cars exist than for example 20 years ago. To own a car has become a basic need in our society in the meantime. Because a large part of the people uses the car as means of transport and because a big part of the transportation of goods is done via the roads the road network is flooded by vehicles. Especially in the rush hours or at the beginning or the end of holidays the road network is overloaded by too many vehicles. This leads to slow-moving traffic and many congestions. The costs of the delay cause an enormous economic damage by both the time that is lost and the higher consumption of petrol. Directly connected with consuming more petrol are the generation of more exhaust fumes and the stronger pollution of our environment.
Consequently many different efforts are undertaken to solve the traffic problems. Large sums of money are invested to extend the road network and the public transport. The opportunities of the internet are more used to allow people to work at home and to communicate with their employer or other colleagues. Also 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 [Eggenkamp 2001].
Since one of the reasons for congestions is that the road capacity is not optimally used another approach is pursued. This is the development of routing systems which use dynamic information about the current state of the road network. By avoiding congested roads and using alternative ways instead the network capacity is exploited in a more optimal way. Cars that use such a routing system are routed around the congestions. And if many cars use alternative roads the congestions will be reduced. Such routing systems route the cars along the path with shortest time and not along the path with shortest distance like nearly all today's navigation systems do. Those conventional navigation systems are nowadays widely used and are offered by many companies. But they are not able to avoid routing a car into congestions because they do not use dynamic data. So if there are congestions on the road network, these navigation systems can only be considered as help for finding the way to a destination but not to find the shortest way in time.
Right now there exist only two applicable navigation systems that use dynamic data. One is the Tegaron Scout from the German Company Tegaron Telematics GmbH [Kroon 2002]. It uses a GSM telephone to connect with the Tegaron station to request routes, which then computes the best route using available dynamic data. The best route is sent back to the Tegaron Scout. But the dynamic data is only available for highways and the costs for using the system are high.
The second is the navigation system VDO Dayton MS 6000 from Siemens VDO (see Figure 1). It calculates the best route on the basis of the latest traffic information by means of integration with messages of the Traffic Message Channel (TMC) [Siemens]. It offers dynamic route guidance, alerting the driver of a problem on the planned route and calculating an alternative route to avoid the incident.

Figure 1: VDO Dayton MS 6000

In [Kroon 2002] a dynamic vehicle routing system was introduced which uses a promising approach, namely the Ant Based Control algorithm (ABC-algorithm). The routing system is able to guide individual car drivers through a city. It always provides the route with shortest travel time and so it avoids congested roads when other roads are faster. But if the congested roads are faster than the alternative paths the car will be routed through the congestion. The ABC-algorithm uses some of the principles that ants in nature use. The algorithm is very suitable for decentralized, dynamic route optimization in networks where the dynamic data changes very fast. So the Routing system is expected to have a great potential compared with conventional route planners. More about this is explained in section 1.2.2.

This thesis develops Kroon's Routing system into a Hierarchical routing system which is not only applicable for a city but also for a larger network with many cities and motorways which connect them. Therefore the Ant Based Control algorithm is adapted to be applicable in a hierarchical network to do a hierarchical routing.

1.2 Hierarchical routing system

The first question that must be considered is: Why do we need a Hierarchical routing system? As it was found out in [Kroon 2002] the size of the network is a limiting factor for the ABC-algorithm. The algorithm does not perform as well for big networks as for small networks. Now when we want to develop a dynamic routing system for a large area with several cities and motorways between the cities it is not possible just to use a larger traffic network for the existing Routing system. The system will need an enormous powerful computer system to function in an adequate speed to be applicable in reality. But even with

such a computer system the size of the network will be very limited. To solve the problem a new approach must be followed. And this is the introduction of a Hierarchical routing system which routes vehicles using a hierarchical network.
The approach is that a separate network exists for every city. And the motorways that connect the cities are part of another network. In a similar way a city itself can be split up into several networks where one network contains the main streets of the city and several smaller networks represent the different areas of the city which are connected by the main streets. In that way the whole network is split up into several smaller networks which can be handled by different distributed computer systems. The idea for computing a route from a street within one city to a street within another city is derived from the humans' procedure. The usual way for doing this is to look at a roadmap first to find a route along the motorway from the environs of the source city to the environs of the destination city. The main reason why this is done is that taking the motorway should be the fastest and easiest route to the destination. And besides, for the sake of efficiency it is not possible to consider every small street that might be useful to take. So humans use a more abstract level first at which only the cities on the road map and the motorways that connect them are considered. After this part of the route is found the city map of the source city is used to find the closest slip road of the chosen motorway. Then the route from the source street to this slip road is planned with the city map. And the same is done for the destination city: The city map of the destination city is used to find a departure of the motorway that is close to the destination street. And from that departure the route to the destination is planned. Therefore a much more detailed level is used, namely the city maps that contain all streets of a city.
To sum up two different kinds of maps are used, one that shows a large area but on a low detail (the roadmap) and other high detailed maps for the city networks which indeed concentrate on parts of the low detailed map (the city maps).
The same approach is used for the Hierarchical routing system to route vehicles to other cities. For the development of the Routing system a simplified traffic network is assumed into which a hierarchy is introduced. This network is defined in Chapter 2.

1.2.1 Dynamic environment

In a static environment the weights of all links of a network are fixed and do not change. The shortest routes for all pairs of nodes need to be computed only once and need not be updated, because the weights do not change. In a dynamic environment, like for a dynamic routing system, the weights of the links are dynamic data which changes over time. By that it is not possible to compute the shortest routes only once because when the weights change also the shortest routes might change. That means that routing in a dynamic environment with dynamic data is completely different to routing in a static environment.
In a dynamic environment the shortest route to a destination can not be computed as a whole beforehand, because during a vehicle follows the route the dynamic data and so the shortest route can change. By that the route to the destination must be computed again regularly to adapt to changed dynamic data.
The dynamic environment can be illustrated with a three dimensional graph like shown in Figure 2. This graph displays four layers at four different times with an interval of one minute between neighboured layers. Every layer shows the same example network consisting of the nodes A to H. The dotted links that intersect the different layers are the

actual travel times to go from the node of the lower layer, where the link starts, to the node of the upper layer, where the node ends. It must be noticed that in Figure 2 only a few links are shown because to show all links would have resulted in a cluttered figure. As can be seen in the graph, the travel times change over time. For example at time t=10:01 the travel time to go from node A to node B is 1 minute. But after one minute, as can be seen in the second layer at the time t=10:02, the travel time to go from node A to node B has changed and is now 2 minutes.

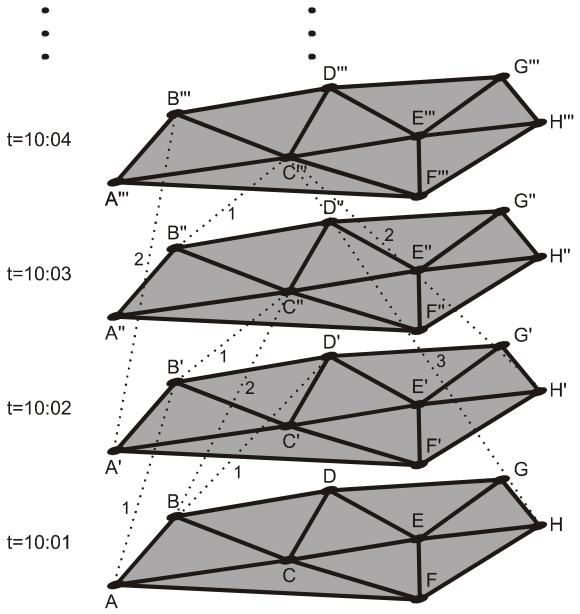

Figure 2: A three dimensional graph

The weights for the links of the traffic network for the Hierarchical routing system, the dynamic data, are the estimated travel times for the vehicles to cover the links. So a short travel time means a low weight for a given link and a long travel time means a high weight.

1.2.2 Decentralized routing

Like the Routing system which was developed in [Kroon 2002] the Hierarchical routing system will use the Ant Based Control algorithm. This algorithm is derived from some principles that ants living in ant colonies use. Ants have a remarkable way to find the shortest way between the nest and a food source. They lay a pheromone trail which is sensed by other ants. In that way they mark different routes to the food source. Among all pheromone trails to the food source the ants are able to find the shortest route after some time. This behaviour is explained in further detail in section 5.2.
Conventional routing algorithms like Dijkstra's algorithm are centralized routing algorithms for a static environment with guaranteed shortest paths. Centralized router assume that all data is available at one location. So they are designed for non-distributed networks and are not suitable for routing in distributed networks. Another drawback is that these algorithms

are in general not suitable for large and dynamic networks because they need a static environment. In a dynamic environment these algorithms have to start all over again to compute the shortest routes for the whole network whenever the dynamic data is refreshed. And the dynamic data might be refreshed several times per minute or even per second. Shortest path algorithms like Dijkstra's algorithm, which work with cost matrices, need for a network with n nodes at least a time of $O(n^2)$ to compute the shortest path from a single source (single-source shortest paths). To compute the shortest paths from every node to every other node the shortest path algorithm must be applied n times. So the resulting time complexity is $O(n^3)$ for the shortest paths of all pairs [Horowitz 1998]. And this is very time consuming for large networks and therefore such shortest path algorithms are not suitable for large dynamic networks.
The ABC-algorithm however is a decentralized routing algorithm and so very suitable for a distributed routing system. It will be able to compute routes also if the data is distributed over several locations. Besides, the ABC-algorithm works with dynamic data without the need to start the computation all over again when the dynamic data changes. But there is no guarantee for finding absolute shortest paths. The ABC-algorithm is explained in detail in section 5.3.

1.2.3 Dynamic data

What has not been considered yet is where to get the dynamic data about the current situation of the traffic network from. There exist mainly three different sources that are currently used to gather information about congestions and up-to-date travel times. These are traffic monitoring systems, emergency services and motorists' calls.

Traffic monitoring systems
Most of the traffic monitoring systems retrieve their information from sensors on the road surface or sensors installed on bridges or traffic information panels. The sensors can count the number of vehicles and measure their speed. From that slow-moving traffic and congestions can be detected. Another source can be cameras installed along the roads which provide visual information about the traffic. Congestions can be identified by humans or using image recognition and the length can be measured. Also the travel times can be gathered with the cameras. Therefore individual cars are watched which are identified by using the license plate. The time which these cars need to go from one position watched by a first camera to another position watched by a second camera is counted. With this time and the locations of the cameras the travel times for that part of the road is measured. A disadvantage of traffic monitoring systems is that they need an expensive infrastructure and so they are not available at every location.

Emergency services
Emergency services can provide information about accidents and the resulting obstruction. For example in the case of a heavy accident on a motorway, where the whole motorway is blocked, the emergency services can send the information to radio stations and traffic information systems that this motorway is closed.

Motorists' calls
Motorists' calls are the third source for information about the traffic flows. This source is available everywhere because the majority of the people has mobile phones. But it is also the most inaccurate and unreliable source because the details that individual drivers can give about the traffic jams are very limited. As a car driver you mostly have not the overview to correctly estimate the length or the delay of congestions.

The information gathered from the traffic monitoring systems, emergency services and motorists' calls are available via traffic news on the radio, via traffic information services like ANWB or Monica and via TMC (Traffic Message Channel). Traffic information services like ANWB provide information about possible congestions on highways via Internet, SMS, WAP and i-mode. Figure 3 shows a roadmap of the area of Rotterdam which was taken from [ANWB] using the Internet. The highways are marked with colours according to the current traffic situation at the different sections. A dark red colour means congestion, light red colours mean slow-moving traffic. White sections are congestion-free. This information is also available in a verbal form. But nevertheless this information is available for the highways only and is not sufficient enough to compute the travel delay for the congested motorways. But exactly this travel delay is needed by dynamic route planners to compute the shortest routes in time.

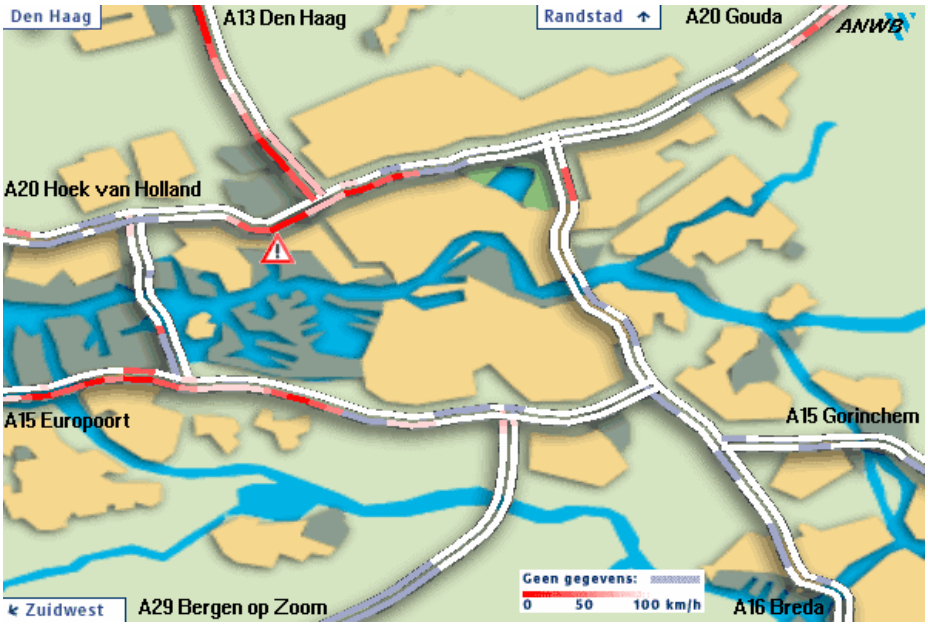

Figure 3: Traffic information provided by ANWB

Another traffic information system is MoniCa, a system that collects and provides dynamic route information for car drivers [Kroon 2002]. It uses digital information panels along the roads to display the dynamic information. By that cars shall be routed in an optimal way. But because information from MoniCa is not publicly available, it can not be used for the Hierarchical routing system.

TMC is a specific application of the FM Radio Data System (RDS) used for broadcasting real-time traffic and weather information [TMCForum]. It is now available in most of the countries in Europe and can be received by a TMC-equipped car radio or navigation system via the normal FM radio antenna. TMC messages can be filtered so that only those relevant to the current journey are processed. Besides the event description and the location also the duration, how long the problem is expected to last, is included in a TMC message. Also a diversion advice is included, whether or not drivers are advised to find an alternative route. If the given information is accurate enough to determine the travel times for congested routes TMC will be suitable for the Hierarchical routing system.

Another source for information about the traffic situation, which has not been considered so far, is tracking vehicles by GPS (Global Positioning System) or GSM. Vehicles that are equipped with a GPS-receiver can be tracked by GPS satellites. Or vehicles with a mobile phone inside can be tracked by the local GSM network. From that information conclusions can be drawn about the speed of the vehicles and the traffic density on the streets. The existing GPS or GSM infrastructure can be used so that no sensors or cameras along the roads are necessary.
But vehicles can also actively provide information about the path they followed and the time they needed to cover it. To locate the current position vehicles can use GPS or GSM. At regular intervals vehicles send the position where they sent the last route information, the current position and the followed path between both positions via a wireless connection to a server. With that information the time estimates to cover each road can be computed on the server. In that way the travel times for every link of the network can be determined from the information provided by the cars. This principle was already used for the Routing system introduced in [Kroon 2002] and is taken over for the Hierarchical routing system. The update process of the dynamic data and the computation of the time estimates for the roads by using update information from vehicles are explained in detail in Chapter 4.

1.3 Project description

The different phases done in this project are enumerated below.

1. Study of dynamic routing systems and routing algorithms.
2. Design of a model for a hierarchical network.
3. Design of a model for the Hierarchical routing system.
4. Adaptation of the ABC-algorithm.
5. Validating the model of the Hierarchical routing system.

1.4 Report structure

This thesis has the following structure:
In Chapter 2 a network model and the way how a hierarchy can be introduced into this network is described. Chapter 3 considers the design of the Hierarchical routing system. In Chapter 4 it is explained how the Timetable updating systems provide the Hierarchical

routing system with dynamic data. Chapter 5 describes the mode of operation of the Route finding systems and the adaptations of the ABC-algorithm. And in Chapter 6 the model of the Hierarchical routing system is validated and recommendations are given.

Chapter 2 : Network model

2.1 Network layout

The kind of network that is considered for the development of a Hierarchical routing system is a simplified traffic network which can be considered as a model of a geographical map of cities, surrounded by rings and connected by highways. An example of such a network is shown in Figure 4. This special example shows a rectangular network consisting of nodes and links, which is divided into several equally sized rectangular sectors. Each sector is surrounded by a closed ring of special links, the motorways, and special nodes, the slip roads, departures or motorway intersections.

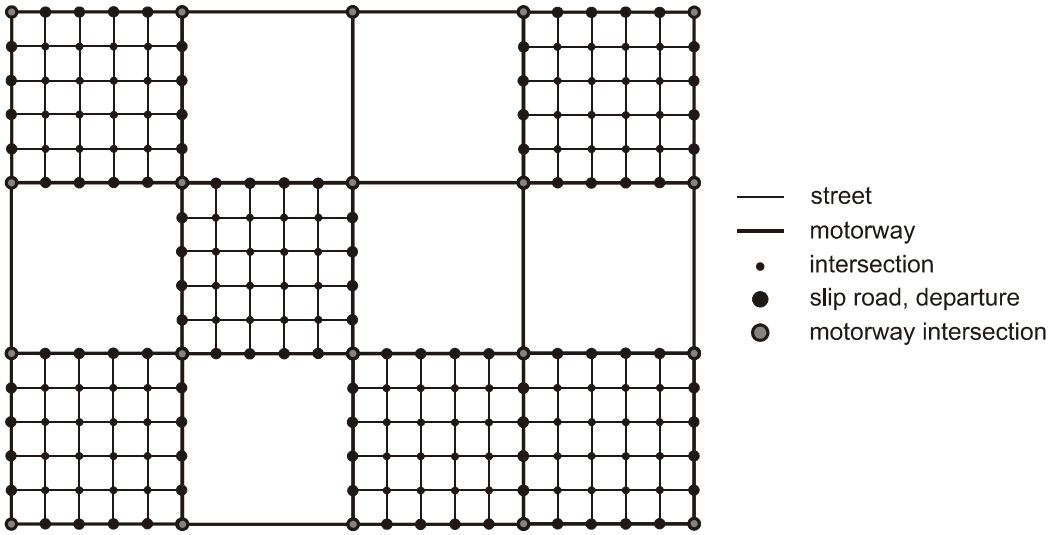

Figure 4: Traffic network example

Inside each sector there can be a further kind of network, a city network, also consisting of nodes and links. These links represent streets in the city and the nodes represent intersections that can be distinguished in different kinds of intersections, such as intersections without traffic lights, intersections with traffic lights or roundabouts.
Slip roads respectively departures are the nodes on the motorways that are linked to a node of a city network (an intersection) by a street. Motorway intersections are the nodes on the motorways that form the edges of a sector. They are only connected with other motorways but not with streets.
Also the type of streets and motorways can be distinguished such as unidirectional streets (on way streets), bidirectional streets, streets with one lane in each direction or with more lanes, motorways with two lanes in each direction, motorways with three lanes in each direction and so on…
For the sake of developing a Hierarchical routing system it is not necessary to go into more detail about the realization of the different kinds of streets, intersections, departures and slip roads on a computer at this moment (this would only be of interest to simulate such a

network). The only interesting property of streets for now is whether a street is unidirectional or bidirectional, what is of great importance for the Routing system.
To realize this distinction the traffic network is a directed graph. It means two directed links exist for every bidirectional street, one for each direction. A unidirectional street has only one link and this is directed to the street's direction. In that principle a slip road will also be a departure if there is not only a street to this node but also a street from this node. And in the same way a departure will also be a slip road if there is not only a street from this node but also to this node.
Additionally the links in the graph are weighted. Here the average time that is estimated to cover a link from the source node to the destination node is used. And these times are the dynamic data, which is provided by the vehicles or sensors and which changes dynamically. Because the links are directed there can be different travel times for a street for each direction, which is the same as in reality.

2.2 Network hierarchy

In the network model introduced in the last section a hierarchy can be introduced by splitting up the network into several less complex networks. The resulting hierarchical network consists of two detail levels, the abstract level and the detailed level. The lower detailed level is the abstract level. On the abstract level there is only one network, the motorway network, which can be compared with a road map. To it belong all motorways, slip roads, departures and motorway intersections. The detailed level consists of several independent city networks, one for each sector that contains a city network. To a city network belong all streets and intersections of the sector and also the surrounding motorway. So a single city network (one sector) of the detailed level can be compared to a city map of the corresponding city. The resulting network hierarchy for the example network of Figure 4 from the last section is displayed in Figure 5.

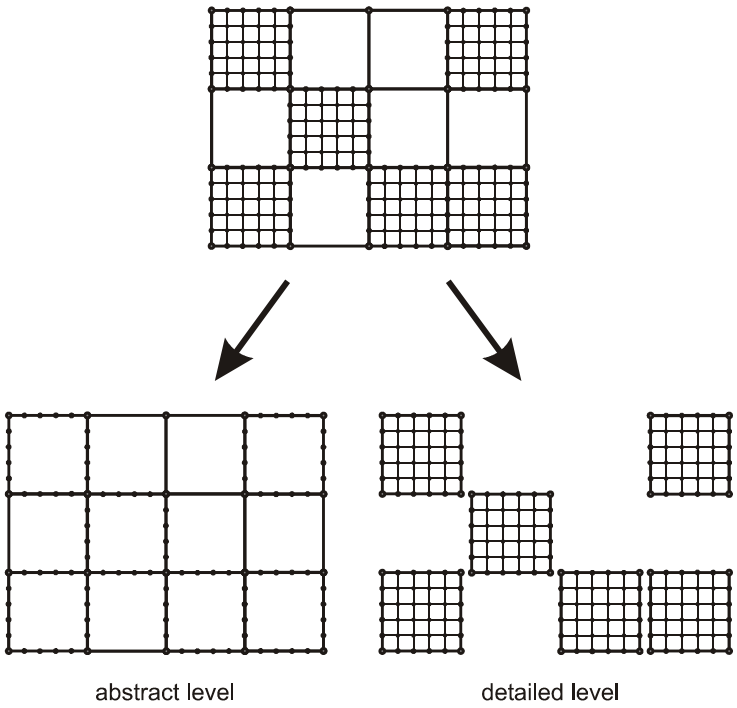

Figure 5: Network hierarchy

By splitting up the network in this way a hierarchical traffic network based on the human's behaviour to plan a route (see section 1.2) is created.

2.3 Internal representation

In this section a possible representation of the above introduced simple network on a computer is explained.
For a Routing system it is essential that all nodes and links of the network have to be identifiable so that they can not be mixed up. Therefore it is necessary to give them a unique identifier. An appropriate way to do this is to number the nodes with natural numbers. The same must be done with the links to be able to distinguish them.
The node's position on the map is defined by nonnegative x and y coordinates. To connect the nodes by the links for every link the link's source and destination, in example the number of the node the link starts from and the number of the node the link goes to, must be defined. Besides, additional information is needed. The kind of node must be defined too. With that information it is clear whether a node belongs to the motorway or to the inner part of a city network. So if a node is defined to be a slip road, departure or motorway intersection, it will belong to the motorway. And if a node is defined to be an intersection it will belong to the inner part of a city network.
Of course some additional information about the links is needed too. For each link the kind of link must be given, whether it is a motorway or a street. And furthermore the links' length and allowed maximal speed for the cars must be defined.

But the most important information needed for a Hierarchical routing system is the sector number to which each of the nodes and links belong to. So the sectors must be numbered with natural numbers and this number must be added to each node and link. The nodes within a city network and the streets are always part of exactly one sector, because they are situated inside a sector. The nodes on a motorway (the slip roads/departures, motorway intersections) and also the motorways as such are mostly part of more sectors. If the nodes are not situated at the border of the network, departures, slip roads and the motorways in the simplified traffic network belong to two sectors whereas motorway intersections, which form the edges of a sector, belong to four sectors. To illustrate this, an example network with six sectors labelled with the numbers 1 to 6 is given in Figure 6. Node A, an intersection, belongs only to sector 1. The slip road/departure labelled with B belongs to the sectors 1 and 2. The slip road/departure C belongs to the sectors 5 and 6. The motorway intersection labelled with D belongs to four sectors, the sectors 2, 3, 5 and 6. But the motorway intersection E belongs only to the sectors 3 and 6 because it is situated at the border of the network. Because of the same reason the slip road/departure F belongs to sector 6 only.

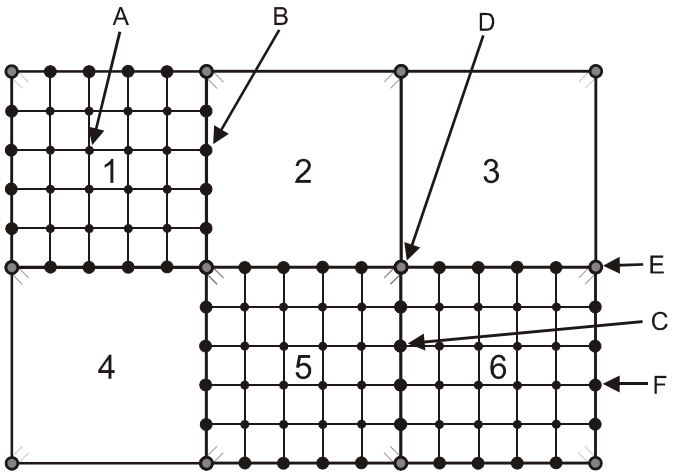

Figure 6: Traffic network with 6 sectors

Given all this information a Routing system is able to create a hierarchical model of the network. It means by using this information the Routing system is able to create an internal representation of the network for the abstract level and internal representations of the detailed level networks for each single sector like displayed in Figure 5.

Chapter 3 : Design

3.1 Conventional design

Now a rough design for a Hierarchical routing system is discussed.
The design of the Routing system introduced in [Kroon 2002] constitutes the starting point of the Hierarchical routing system. Many points of that design can be taken over without changes and others need to be adapted to the hierarchy.
Let's call this design to mind. Figure 7 shows the design of the Routing system introduced in [Kroon 2002]. The first remark that must be mentioned when explaining this figure is that the Routing system as such is not placed in the vehicles but runs on a possibly distributed computer system. Vehicles which use the Routing system connect via a wireless connection. In the vehicles a handheld computer or a dedicated navigation system is used. This device serves as interface between the driver and the Routing system. It must enable the driver to enter a destination and to display the route to be followed which is computed by the Routing system. Furthermore the device must be able to derive a position on a road by using GPS (Global Positioning System). The device must have the graph of the whole traffic network in its memory. This enables the device to display the route and to know at what position in the network the vehicle currently is. All the communication between the vehicle and the Routing system like explained in the following goes over this device.

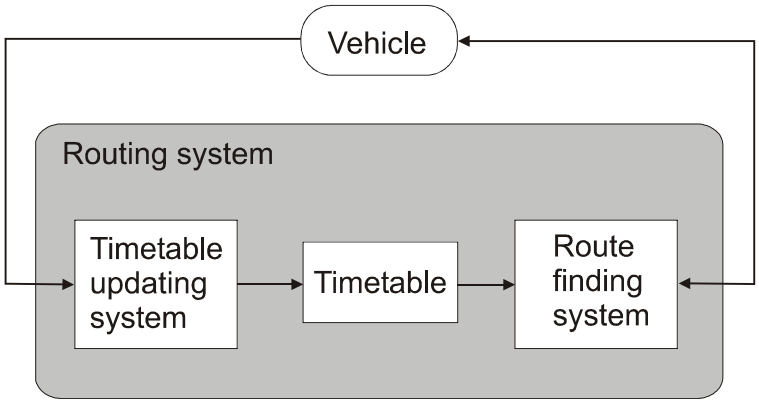

Figure 7: Design of the Routing system

The Timetable updating system and the Route finding system together form the Routing system. The Route finding system is responsible for routing vehicles from a source to a destination along the shortest route in time. For that the vehicles make a request for a route to the Route finding system. The Route finding system uses the dynamic data in the timetable to optimize the routes from every node in the network to every other node by using the ABC-algorithm. It receives the vehicles' route request, takes the concerned optimized route and sends them back to the vehicles. The timetable is a kind of memory which contains dynamic data about the state of the network measured in time that is needed to cover each single link from its source to its destination. It is a two-dimensional matrix with all nodes of the network graph along the axes. In this table an entry exists for each link

of the network. And the entry represents the time estimate for the vehicles to cover the corresponding road. Table 1 is the timetable for the example network shown in Figure 8. In this example the estimated time to go from node 1 to node 2 is 9 seconds. And for example the estimated time to go from node 2 to node 1 is 8 seconds.

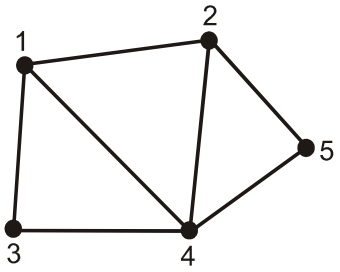

Figure 8: A simple traffic network

Table 1: The timetable from the network in Figure 8

From ↓ | To →	node 1	node 2	node 3	node 4	node 5
node 1		9s	11s	12s	
node 2	8s			8s	6s
node 3	10s			7s	
node 4	13s	9s	8s		9s
node 5		6s		10s	

The task of the Timetable updating system is to fill and update the timetable with time values according to the current state of the traffic network. It means that the Timetable updating system provides the Route finding system with dynamic data. It receives data from vehicles about the time needed to travel from a certain source to a destination. With that data the estimate for the time to cover each road the vehicle drove along is computed and updated in the timetable.
Note that it is not assumed that every vehicle of the traffic is equipped to use the Routing system. There is a part of the vehicles which has the necessary hardware to use the Routing system, but the other part not. It is desirable that the equipped vehicles all provide the Timetable updating system with update information to have as much dynamic data as possible. But not every driver might want to use the Routing system to be routed. For example buses might provide update information, but they need not be routed because they always drive along a fixed route. What is very important to remark is that the more vehicles use the Routing system, the higher the effect to avoid or reduce congestions will be, because more vehicles are routed around congestions.

3.2 Extended design

Now that we are interested in a Hierarchical routing system for the sake of handling more complex networks this design must be adapted.

Figure 9 shows the adjustments.

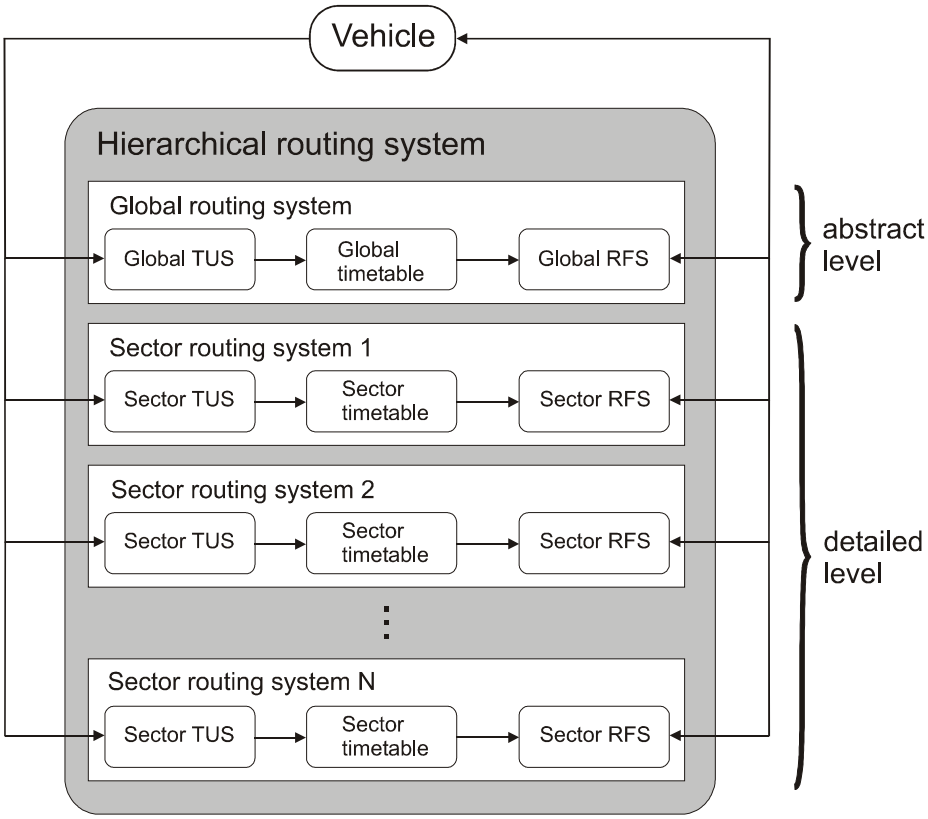

Figure 9: Design of the Hierarchical routing system

The Hierarchical routing system consists of several parts. These are the Global routing system and a Sector routing system for each sector that contains a city network. So if there exist N sectors with a city network then the Hierarchical routing system will consist of N Sector route finding systems. Every part of the Hierarchical routing system consists of the same components as shown in Figure 7, namely a Timetable updating system (TUS) and a Route finding system (RFS). Besides all Routing systems have an own timetable which stores the dynamic data of the traffic network.

The handheld computer or dedicated navigation system in the vehicles must have the internal representation of the whole network as explained in section 2.3 in its memory. This enables the device to know the type and the sector of every node and every link of the traffic network. By that the device knows for every position of the vehicle whether it is a position on a motorway or in a city network. And it always knows the sector where the vehicle is in. That information is very important for the device to know what part of the Hierarchical routing system is responsible to route the vehicle and to what part the update information must be sent.

3.2.1 Global routing system

The Global routing system is responsible for the abstract level of the traffic network (see Figure 5). Its Global route finding system computes the shortest routes in time for the vehicles from a node of the motorway to another node of the motorway along the motorway only. Therefore it uses the global timetable which holds the dynamic data of the abstract level of the network. It means for each motorway link of the network there is an entry in the global timetable which represents an estimate for the time to cover that part of the motorway. To keep this data up to date the Global timetable updating system updates the global timetable by using update information from vehicles that use the motorway.
The Global routing system only works with the internal representation of the abstract level of the traffic network and not with the networks of the detailed level. But nevertheless the Global routing system must have the information to what sector all the other nodes of the traffic network belong, which are not nodes of the abstract level.

3.2.2 Sector routing system

A Sector routing system is responsible for one sector of the detailed level which contains a city network (see Figure 5). Therefore it only works with the internal representation of the city network (detailed level) of that sector. Additionally a Sector routing system needs the information for all other nodes of the traffic network to what sector they belong and what type of node they are (intersection, slip road/departure or motorway intersection).
A Sector routing system routes vehicles within the corresponding sector from one position of the sector to another position of the sector along the shortest route in time. For that all streets of the sector and also the surrounding motorway ring can be used to reach the destination. To compute the shortest routes a Sector route finding system uses the corresponding sector timetable. A sector timetable holds the dynamic data for all streets within the corresponding sector. The Sector timetable updating system receives update information from vehicles inside the sector about the route and the time to go from one position in the sector to another position in the sector. With that data it can compute the time estimates for the covered streets of the sector and update them in the sector timetable. Additionally a Sector route finding system needs the time estimates for the motorway links which surround the sector, because the motorway ring can also be used to route vehicles to another destination in the sector. The time estimates for the motorway links are computed by the Global timetable updating system and are already available in the global timetable (see section 3.2.1). Because of the communication delay it is not desirable to send a request to the Global routing system every time a time estimate of the global timetable is needed. This would considerably slow down the performance of the ABC-algorithm. Therefore the time estimates of the surrounding motorway links should also be stored locally in the sector timetables, what allows a fast access to these values. More details about this will be given in section 4.5.

3.2.3 Cooperation

The Global route finding system and the Sector route finding systems can operate independently to calculate routes within the area they are responsible for. But if a vehicle requests a route for instance from a street within one sector to a street within a distant sector several Route finding systems will have to cooperate to compute the shortest route in time. The approach described in section 1.2 serves as pattern for this cooperation:
To reach a street in a distant sector the car should use the motorway to go from the source sector, where the car is actually in, to the destination sector, where the street is in. So the car must reach the motorway first. The only points along whose a vehicle can leave a sector in the assumed simplified traffic network (see Figure 4 in section 2.1) are the four motorway intersections. The aim is to route the vehicle as fast as possible to the destination sector. So the Sector route finding system of the source sector must route the vehicle to the motorway intersection along which the time to reach the destination sector is minimal. The part of the route from that motorway intersection to the destination sector is computed by the Global route finding system. And the last part of the route from the motorway intersection of the destination sector, where the vehicle arrives, to the destination node is computed by the Sector route finding system of the destination sector.
How this cooperation works in detail is explained in Chapter 5.

3.2.4 Distribution and communication

The design of the Hierarchical routing system introduced in Figure 9 allows the system to be distributed among several computer systems. The Global routing system and the Sector routing systems each can run on a separate computer system. These computer systems must be mutually connected via a network to cooperate as a common system. The vehicles use a wireless connection to connect with the Hierarchical routing system. Compared with a Routing system that runs on one computer system the distribution has several advantages. Because now several computer systems fulfil the task that one computer system did, the speed and available memory space increases. Besides, the failure of one of the computer systems does not have to imply a total breakdown of the whole Routing system. The drawback of the distribution is that always some communication between the distributed computer systems is necessary.
Now the question is where to place each Routing system in a real traffic network. There exist mainly two different approaches.
The first is a centralized arrangement of all distributed computer systems at one point of the traffic network. The computer systems can be connected via a local area network (LAN). On the one hand a centralized arrangement means a very short distance for the communication between the parts of the Hierarchical routing system and very short response times. But on the other hand the average, wireless communication distance between the vehicles and the centralized Hierarchical routing system is very long what means a high communication delay and high response times.
The second approach is a decentralized arrangement of the distributed computer systems. Every computer system is locally placed close to the area it is responsible for. The computer

systems can be connected via a wide area network (WAN) or the Internet. Let's assume that the cars use a direct wireless connection with the responsible part of the Hierarchical routing system. Therefore the communication of the vehicles with the responsible Routing system goes over a short distance and does not cause such a high delay. Now the communication between the distributed parts of the Hierarchical routing system goes over a long distance. But because the Global routing system is responsible for the abstract level which extends over the whole network it can not be placed locally. So the second approach is not applicable for the Hierarchical routing system.
Nevertheless, a mix out of both approaches is feasible and at the same time the best solution. This approach is displayed in Figure 10.

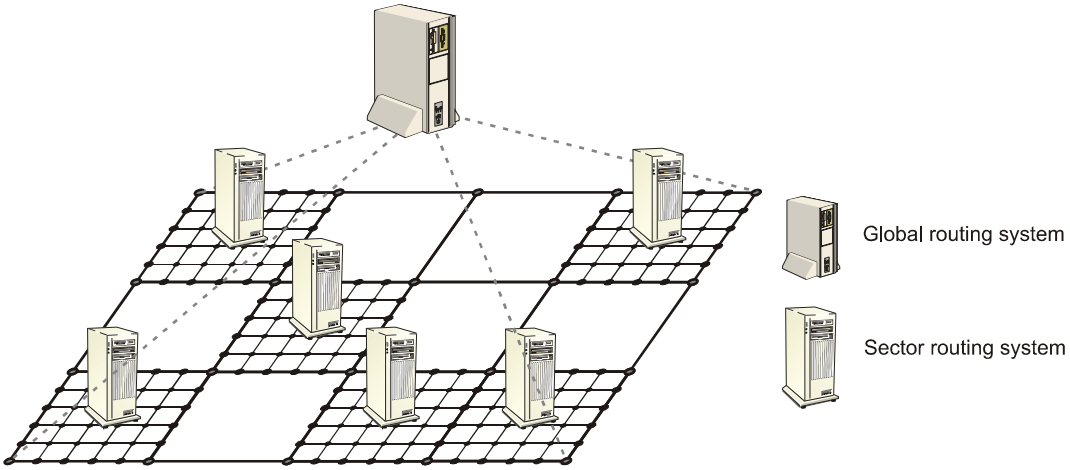

Figure 10: Distribution of the Hierarchical routing system

The Sector routing systems which are responsible to route vehicles within a sector are situated within or at least close to the sector they are responsible for. The Global routing system is centralized because it is responsible for the whole motorway network. For the communication between the Routing systems the Internet or a wide area network is used. A connection must exist from every Routing system to every other Routing system to work as a common system. The connections for a Hierarchical routing system with four Sector routing systems are displayed in Figure 11.

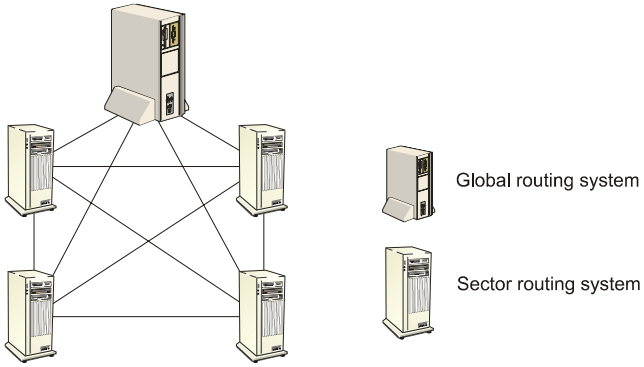

Figure 11: Communication between the distributed parts

Vehicles that want to use the Routing system directly connect with the responsible part of the Routing system using a wireless connection, for example GSM.
The following rules define with what part of the Routing system vehicles must connect:

- A vehicle on a street inside a sector which wants to request a route must connect with the Sector route finding system of that sector. (I)
- A vehicle on a street inside a sector which wants to send update information must connect with the Sector timetable updating system of that sector. (II)
- A vehicle on a motorway which wants to request a route must connect with the Global route finding system. (III)
- A vehicle on a motorway which wants to send update information must connect with the Global timetable updating system. (IV)

Figure 12: Rules for the connection with the Hierarchical routing system

Besides, vehicle must connect with a GPS-satellite to determine their position. The position is needed for requesting routes as well as for sending update information. So the connection with a GPS-satellite to determine the position is always done before the vehicle connects with the Hierarchical routing system.
Figure 13 displays the communication of the vehicles with GPS-satellites and the direct, wireless connection with the Hierarchical routing system according to the rules defined in Figure 12. Note that the connection of the vehicles with the Hierarchical routing system is not synchronized, that is the vehicles do not synchronously connect with the system at the same time to request a route or to send update information. Also the communication with the GPS-satellites is not synchronized.

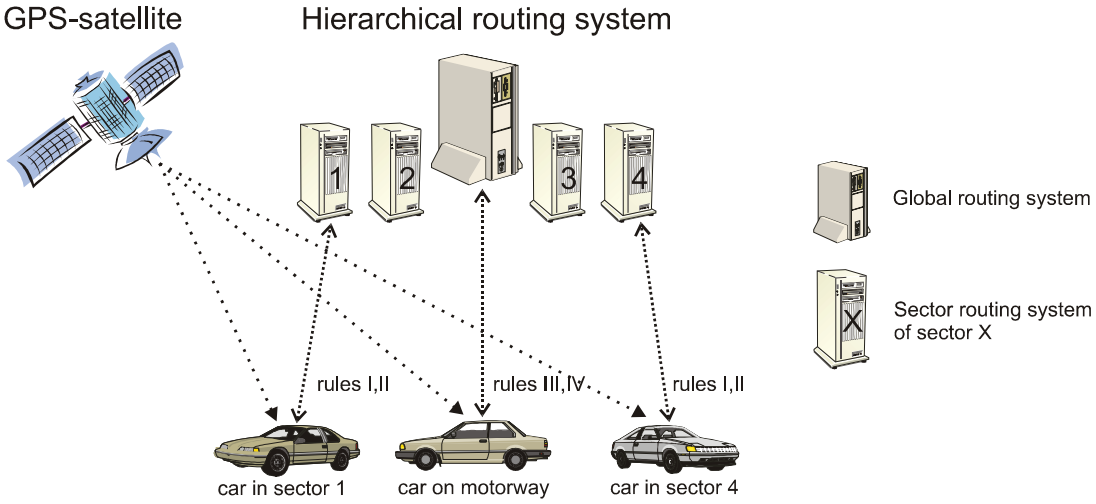

Figure 13: Vehicle connection

By the direct connection of the vehicles with the responsible part of the Hierarchical routing system and the chosen distribution it is ensured that vehicles, that need to be routed in a city, receive the route information over a short communication distance with a low delay only. This is very important because the distance between two intersections in a city is very short. And for example a car, which requests a route update at every intersection, needs the updated route very quick to know where to go at the next intersection. The communication between cars and the Global routing system goes over a longer distance. But since on a motorway the distance between slip roads/departures or motorway intersection is much longer than the distance between two intersections in a city, a higher communication delay should be problem-free.
Nevertheless it is possible to overcome with the high communication delay by distributing the Global routing system too. This is possible because of the usage of the ABC-algorithm which works with distributed networks. Therefore the abstract level is divided into sections. For each section there must be one part of the distributed Global routing system that cares about this section. Cars on a motorway connect to the part that is responsible for the section the car is in. And this should be the closest one so that the communication goes over a short distance only.
The same can be done with the Sector routing systems. So if there are large sectors with many streets and heavy traffic, it will also be necessary to divide the sectors into sections and distribute the Sector routing system according to the sections. But in the following a distributed Hierarchical routing system with a non distributed Global routing system and non distributed Sector routing systems like displayed in Figure 10 is assumed.

Chapter 4 : Timetable updating system

A Timetable updating system receives update information about the current traffic situation from possibly different sources from the low sensoric level up to the high semantic level (see section 1.2.3). It uses the data to compute travel times for every link of the network. With the travel times a Timetable updating system updates the time estimates in the timetable.

4.1 Update information

In principle several different sources can be used to get dynamic information about the current situation in the traffic network like explained in section 1.2.3. Up to now the Timetable updating systems of the Hierarchical routing system uses only the vehicles that provide the system with update information. That is the easiest way because the information from traffic information services or TMC can not directly be converted into travel times for the links.
The vehicles send update information at regular intervals about the last part of the path they followed and the time it took them to cover it. With the update information of the vehicles a Timetable updating system can compute the travel times for every covered link and use these times to update the corresponding time estimates in the timetable.
The data structure of the update information, which vehicles send to a Timetable updating system, contains the location and time at the moment of the previous update, the location and time at the moment of the update and all the links the vehicle has covered between both positions. Besides, the vehicles must also provide the times when the motorway was entered, the motorway was left respectively a new city network was entered. This is necessary because in the Hierarchical routing system exist a Global timetable updating system and several Sector timetable updating systems which update their own timetable. And because vehicles do not always go within one sector or on the motorway only the update information which the vehicles record can concern several Timetable updating systems. To be able to split the update information according to the responsibility of the different Timetable updating systems, the times, when the motorway was entered, the motorway was left respectively a new city network was entered, are needed.
The positions where the updates are done are determined by using GPS. For determining the covered links and the times when the vehicle entered a motorway, left a motorway or entered a new city network, the vehicle must also know its current position during it travels. Two different solutions are possible for the vehicles to track their position during two updates. Like navigation systems do, sensors of the vehicle can be used to compute the position. The sensors measure the wheel revolution, the steering angel and the direction of the vehicle. With the known position derived from GPS and the data from the sensors a vehicle can compute and track its position. In that way the vehicle can determine its position on the map and record the covered links and the times when it entered the motorway, left the motorway or entered a new city network. The second solution how a vehicle can track its location between two updates is to request the position from GPS-satellites more frequently.

So GPS is not only used to determine the position at the locations where the update is sent but also between them.

The rules (II) and (IV) of Figure 12 in section 3.2.4 define with what Timetable updating system a vehicle must connect if it wants to send update information to the Hierarchical routing system. Whenever a vehicle is on a street inside a sector, rule (II) applies. So the vehicle must connect with the Sector timetable updating system of that sector. When a vehicle is on a motorway, rule (IV) applies and the vehicle must connect with the Global timetable updating system.

Let's consider some examples to illustrate the data contained in the update information and the connection with the responsible Timetable updating system:
There exist two cases where the update information concerns only one Timetable updating system. The first one is the case where a vehicle has driven on the streets inside a sector only since the last update. At the moment of the update the car is still inside this sector but not on a motorway and therefore it connects with the Sector timetable updating system of that sector. Figure 14 shows an example of this case. It displays the route of a vehicle inside a sector that sent the last update at position 1 at the time of 8:43:15. The vehicle covered the links 1, 2, 3, 4 and 5. At position 2 at the time of 8:43:55 the vehicle makes a new update. It connects with the Sector timetable updating system of the sector and sends the update information consisting of position 1, the time 8:43:15, position 2, the time 8:43:55 and the links 1, 2, 3, 4 and 5.

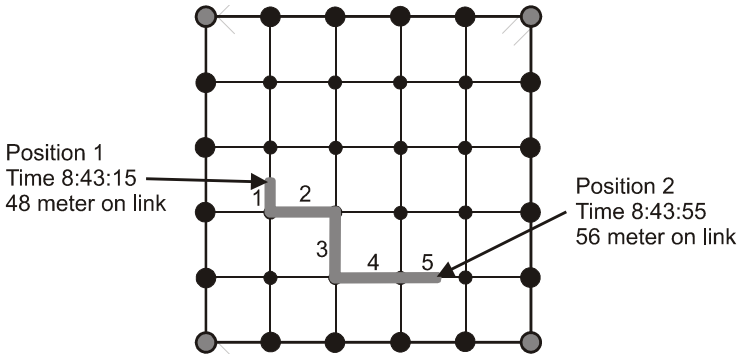

Figure 14: Route within a city network

The second simple case, where the update information concerns only one Timetable updating system, is the case where a car has driven on a motorway only since the last update. So the car is actually on the motorway and it sends the update information to the Global timetable updating system. An example of that case is displayed in Figure 15.

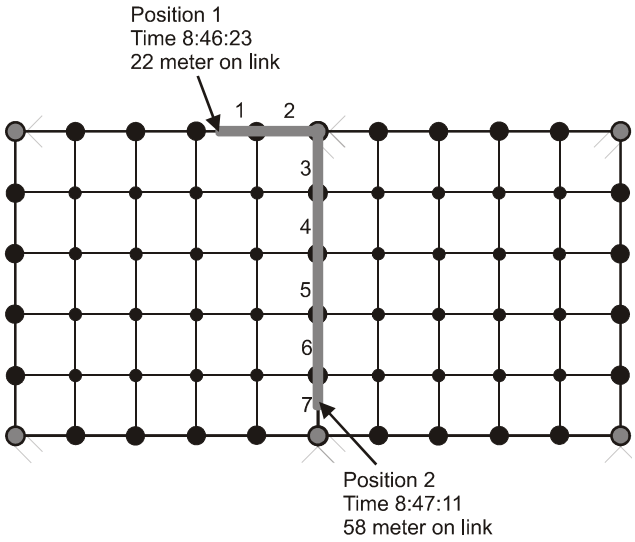

Figure 15: Route on a motorway

Besides the cases where only one Timetable updating system is involved exist also cases that are more complicated. These are the instances where vehicles have not used the motorway only or only the streets within one sector without the motorway since the last update. Now two or even more parts of the Timetable updating system are involved. For example a car might have gone from a street within one sector to a slip road and then along the motorway. Two different parts of the Timetable updating system must be informed to update their timetables. These are the Sector timetable updating system of the sector and the Global timetable updating system.
But also instances can occur where for example three timetables must be updated. One possibility for that is a car starting again within one sector, using the motorway and then entering another sector. An example for that is shown in Figure 16.

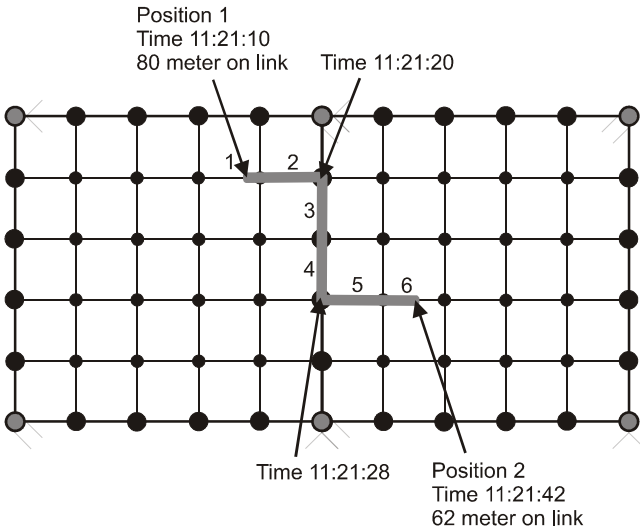

Figure 16: Route within 2 sectors

When the vehicle is at position 2 it sends the update information of the route it has followed since the last update at position 1. Position 2 is a position inside a sector. So the vehicle connects with the Sector timetable updating system of that sector. The update information consists of position 1, the time 11:21:10, position 2, the time 11:21:42, the links 1, 2, 3, 4, 5 and 6 and the times 11:21:20 and 11:21:28. The time 11:21:20 is the time when the motorway was entered and the time 11:21:28 is the time when the motorway was left and a new city network was entered.

4.2 Splitting the update information

In the cases where the update information concerns two or more Timetable updating systems the update information must be split by the Timetable updating system which receives the update information. The points at which the covered route is split into parts are the points in the network where the responsibility of a Timetable updating system ends and a new Timetable updating system is responsible. These are the transitions from the streets of a sector to the motorways, from the motorways to the streets of a sector and from the streets of a sector to the streets of a neighboured sector by crossing the motorway. With the information about the covered route contained in the update information of the vehicles these points are identifiable because the sector and type of every link is known. And with the times in the update information, when the motorway was entered, the motorway was left or a new city network was entered, the times when a vehicle reached these points are known. So the update information can easily be split into parts according to the responsibility of the Timetable updating systems. The resulting, split parts of the update information must be sent to the responsible Timetable updating systems by the Timetable updating system which received the update information and split it.

How the splitting of the update information is done is explained with the example from Figure 16 from the last section:
The Sector timetable updating system of the right sector receives the update information from the vehicle. It must split the route information into three parts because three different Timetable updating systems are involved. These are the Sector timetable updating system of the left sector, the Global timetable updating system and the Sector timetable updating system of the right sector. The points where the route must be divided in this example are the point where the motorway is entered (the node between links 2 and 3) and the point where the motorway is left (the node between links 4 and 5).
The first part of the split update information consists of position 1, the time 11:21:10, the node between the links 2 and 3, the time 11:21:20 and the links 1 and 2. This update information is sent to the Sector timetable updating system of the left sector by the Sector timetable updating system of the right sector.
The second part consists of the node between links 2 and 3, the time 11:21:20, the node between the links 4 and 5, the time 11:21:28 and the links 3 and 4. That update information is sent to the Global timetable updating system by the Sector timetable updating system of the right sector.
The third part consists of the node between the links 4 and 5, the time 11:21:28, position 2, the time 11:21:42 and the links 5 and 6. That part remains at the Sector timetable updating system of the right sector and is processed by this system.

In that way update information, which concerns more than one Timetable updating system, is split and sent to the other responsible Timetable updating systems.

4.3 Computing the travel times

Here in this section it is explained how the update information is used to compute the travel times for every link of the covered route. The travel times for every link are needed by the Timetable updating systems to update the time estimates in their timetables. Because only the start time and the end time of the routes is given in the update information, only average travel times can be computed for every link. For this section it is assumed that in cases, where the update information concerns more than one Timetable updating system, the update information was already split like explained in the previous section and so it is available at the responsible Timetable updating system and concerns only this system.

Now let's consider how the computation of the time for every single link of the covered route can be done:
First of all, the total time to cover the route without delay at intersections or on the links needs to be calculated. The formula for this is:

$$T = \sum_{l} \frac{d_l}{S_l} \tag{1}$$

With the optimal time T calculated with formula (1) the travel time for every single link can be computed by using the following formula:

$$M_l = \frac{L_l}{S_l \cdot T} \cdot t \tag{2}$$

- d_l is the covered distance on link l in meters
- L_l is the length of link l in meter
- S_l is the allowed speed on link l in meter per second
- T is the total time in seconds that is needed to cover the route without delay
- t is the total time of the covered route in seconds. $t = t_2 - t_1$
  with t_1 start time and t_2 end time
- M_l is the measurement of the travel time for link l in seconds

With the computed travel times M_l the timetable of the Routing system, where the Timetable updating system belongs to, is updated. The updating process of the time estimates in the timetables is explained in detail in [Kroon 2002] in section 3.4.1.
The two formulas that were introduced in this section take the allowed speed on the links into consideration. It makes the computed travel times more accurate than the ones computed with the formula in [Kroon 2002] that does not consider different speeds but only the length of the links.

To illustrate the usage of the formulas (1) and (2) an example is given. Imagine a car driving within a city network like shown in Figure 17. In this example network all links have a length of 100 meter. A speed of 30 km/h for link 3 and 50 km/h for all other used links is given.

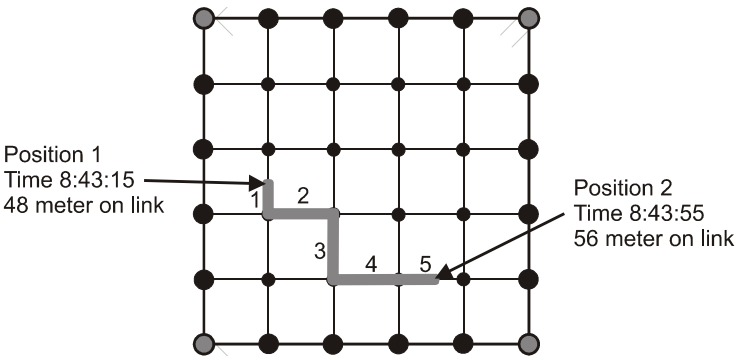

Figure 17: A car's route within a city network

The update information, which the car sends to the Sector timetable updating system of the sector at position 2, consists of position 1, the time 8:43:15, position 2, the time 8:43:55 and the links 1, 2, 3, 4 and 5. The car needed 40 seconds for this route. With the update information the Timetable updating system can compute the travel times for the separate links by using the two formulas:

T is computed with formula (1) and has the following value:

$$T = \sum_{l} \frac{d_l}{S_l} = \frac{52m}{18\frac{m}{s}} + \frac{100m}{18\frac{m}{s}} + \frac{100m}{10{,}8\frac{m}{s}} + \frac{100m}{18\frac{m}{s}} + \frac{56m}{18\frac{m}{s}} = 26{,}4s$$

This is the optimal time to cover the route without any delay at intersections or on the links. The travel times M_2, M_3 and M_4 for the links 2 , 3 and 4 are computed with formula (2):

$$M_2 = \frac{L_2}{S_2 \cdot T} \cdot t = \frac{100m}{18\frac{m}{s} \cdot 26{,}4s} \cdot 40s = 8{,}4s$$

$$M_3 = \frac{L_3}{S_3 \cdot T} \cdot t = \frac{100m}{10{,}8\frac{m}{s} \cdot 26{,}4s} \cdot 40s = 14s$$

$$M_4 = \frac{L_4}{S_4 \cdot T} \cdot t = \frac{100m}{18\frac{m}{s} \cdot 26{,}4s} \cdot 40s = 8{,}4s$$

The travel times of 8,4 seconds for links 2 and 4 and 14 seconds for link 3 reflect the fact that more time is needed for the same distance on a street with an allowed speed of 30 km/h than on a street with an allowed speed of 50 km/h.

Hence M_2, M_3 and M_4 are computed, these values are used to update the timetable of the Sector routing system.

4.4 Update interval

The computed travel times computed by the formulas (1) and (2) are only average values and can strongly deviate from the actual travel times of the cars. For example one of the streets can be congested and others are congestion-free. Then the delay of the congestion on one of the links is distributed over the computed travel times of all links of the covered route although only one street is congested. And this results in falsified travel times. Another source for falsified times are car driver that stop to do some shopping, to tank or to do something else. Then the resulting times do not reflect the current situation of the traffic. The deviation of the computed travel times from the real travel times can be reduced if the vehicles send update information at every intersection. So really the time is given which is needed to cover one link of the network. But a drawback is the amount of data and communication required between the vehicles and the Timetable updating systems. To reduce the communication a longer update interval is desirable. But a too long interval means that the vehicle covered many links till it send the update information. So the travel times are computed as average over all these links and furthermore the delay between driving on a link and updating the time estimate of that link might get too long.
Therefore a compromise must be found for a suitable update interval to avoid too much communication on the one hand and to ensure accurate and actual travel times on the other hand.

4.5 Distributing the global timetable

The time estimates for all motorway links are available in the timetable of the Global routing system (global timetable) as explained in section 3.2.1. But also the Sector routing systems need time estimates for motorway links, namely for the motorway links of the surrounding motorway ring. That is because the motorway rings also belong to the city networks which they surround and can be used by the Sector routing systems to route vehicles within the sector. The time estimates in the global timetable of the surrounding motorway ring for each city network should also be stored locally at the Sector routing systems. This is advisable to guarantee the Sector route finding systems a fast access to the time estimates. An access to these values from the Sector routing systems to the Global routing system via the internet or a wide area network always involves a considerable communication delay. And this must be avoided to secure an optimal performance of the ABC-algorithm. By providing a local copy at the Sector routing systems a fast access can be guaranteed.
Now an example is given to illustrate what time estimates of the global timetable must be provided at what Sector routing system. Figure 18 shows a small traffic network which consists of two neighboured city networks and surrounding motorway rings. A Hierarchical routing system for this network consists of the Global routing system and two Sector routing systems. The Global timetable updating system receives update information from vehicles

that use the motorway. It updates the time estimates in the global timetable. The global timetable contains the time estimates for all motorway links of the network, in the example these are the links 1 till 35. The timetable of the Sector routing system of sector 1 contains the time estimates for all streets of sector 1 but not for the motorway links. And the timetable of the Sector routing system of sector 2 contains the time estimates for all streets of sector 2 without the motorway links. In this example the Sector routing system of sector 1 needs a local copy of the time estimates of the links 1 till 5, 11 till 20 and 26 till 30 from the global timetable. And the Sector routing system of sector 2 needs a local copy of the time estimates of the links 6 till 10, 16 till 25 and 31 till 35.

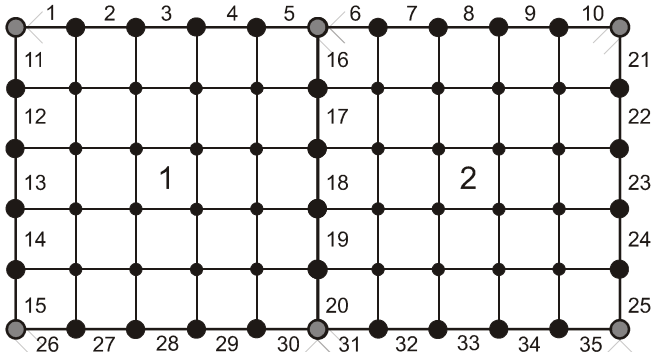

Figure 18: Traffic network with two city networks

The time estimates in the timetables change very quickly, possibly several times a second. For the Route finding systems it is essential to have access to the actual time values all the time. As soon as time values change the changed values should be available. Therefore changes of the time values in the global timetable must be taken over to the locally stored copies in the Sector routing systems with a very short delay only. That means that the update rate for the copy process must be very high. Since the time estimates in the timetables are not supposed to oscillate strongly but to adapt step by step to new traffic situations, these values do not dramatically change with each update. This helps to reduce the amount of data which must be copied to the Sector routing systems at every update step and therefore it ensure a high update rate. A way to reduce the amount of data is to introduce an update measure. The update measure has a numeric value which represents a change in per cent. At every update step a time value of the global timetable is only copied to the Sector routing system if it has changed more than the value of the update measure in per cent since it has been copied the last time. A convenient value for the update measure can be 5. Then a time value is only copied again if it has changed more than 5% since it has been copied the last time. In that way not all time estimates of the Global timetable must be copied to the Sector routing systems at every update interval. And this results in an economy of data which is sent at every update step.

Chapter 5 : Route finding system

A Route finding system is the part of the Routing system that computes the shortest routes in time for vehicles from one position in the network to another position. Vehicles send a request to it to ask for a route from their current position to a destination.

5.1 Design

The Route finding system can be divided into two modules as can be seen in Figure 19, the Route optimization module and the Route computation module.

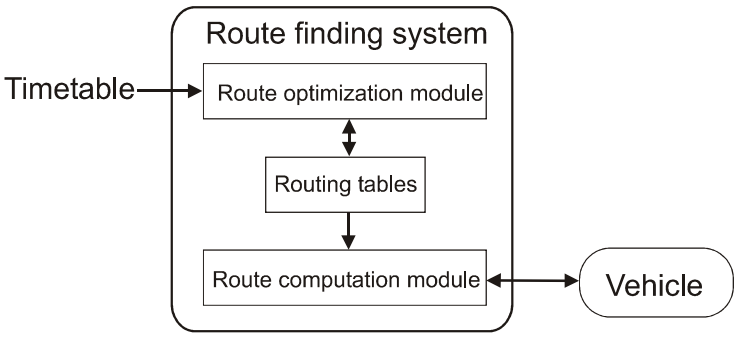

Figure 19: Structure of a Route finding system

The Route optimization module is responsible to optimize the routes from every node of the network to every other node. Therefore it uses the dynamic data in the timetable of the Routing system. For the optimization the Ant Based Control algorithm (ABC-algorithm) is used. The ABC-algorithm stores the information about the shortest routes in routing tables.
The second module, the Route computation module, is the part of the Route finding system that computes the shortest routes in time for vehicles from a source node to a destination node. Vehicles connect with this module to request a route. The Route computation module uses the information in the routing tables, which is provided by the Route optimization module, to compute the routes. After the routes are computed they are sent back to the vehicles.

The ABC-algorithm was derived from the natural behaviour of ants living in ant colonies. The behaviour, which gives the ants the ability to optimize routes, is explained in the next section, section 5.2. How routes are optimized with the ABC-algorithm in general and how routes are computed by using routing tables is explained in section 5.3. But to be applicable for the Hierarchical routing system to work with a hierarchical network, the ABC-algorithm and the computation of the routes must be adapted. These adaptations are explained in section 5.4, which constitutes an important part of this thesis.

5.2 Ant Colony Optimization

Ant Colony Optimization belongs to the new research field swarm intelligence. In swarm intelligence the swarm behaviour of species like ants, bees, wasps, termites, fish and birds is investigated. All these species are individuals with a quite simple structure. But together in a swarm they can achieve a very complex collective behaviour by interacting, which allows them to solve a global objective in a more efficient manner than one simple individual could do. The behaviour of the swarm is closely linked to the behaviour of the single individuals. On the one hand the collective behaviour of the individuals determines the behaviour of the swarm. On the other hand the behaviour of the swarm determines the conditions under which the individuals perform their actions. The social interaction between the individuals is essential for the collective behaviour. By that the knowledge about the environment is improved and therefore the behaviour of the swarm is optimized.
Social interaction between the individuals can be direct or indirect. Direct interaction is the directly visual, audio or chemical contact between individuals. Indirect social interaction happens when one individual changes the environment and the future behaviour of other individuals is influenced by the changed environment. This type of interaction is called stigmergy.
Let's consider the optimization in ant colonies now. The peculiarity of ant colonies is that they consist of a large number of individuals in which different morphological types exist which are fitted to fulfil certain specialized tasks. Collectively the ants fulfil complex tasks like the building of optimal nest structures, protecting the queen and the larva, cleaning the nest, finding the best food sources and optimizing the attack strategies. These activities occur distributed without a central command centre. They are made possible through indirect interaction (stigmergy). This means that the behaviour of the ants is only determined by reacting to local stimuli of the environment. The ants can change the local stimuli. By that the future actions of other ants at this location are influenced.
Two different ways to change stimuli can be distinguished. On the one hand the physical characteristics of the environment can be changed for example by digging a hole. By doing this other ants may be caused to enlarge the hole. This type is called sematectonic. On the other hand the ants can change the local stimuli and therefore the behaviour of other ants at this location by dropping pheromones. This is called sign-based stigmergy.
An example for sign-based stigmergy is the finding of the shortest path by ants between the nest and a food source. This ability of the ants was proved and investigated in several experiments. As the researcher found out the ants looking for food or returning to the nest tend to follow the path with the strongest pheromone trail. So at every intersection they check where the trail is strongest and with a high probability they continue to move along this path. Nevertheless some ants do not follow the paths with strongest pheromone trail, but they follow paths with less pheromones or even paths without a pheromone trail. This ensures that the ants explore new paths to investigate alternatives.
While the ants move they drop further pheromones on the ground by stopping briefly and touching their gaster which carries the pheromone secreting gland. The strength of the pheromone trail they lay depends on the dropping rate and the amount per deposit. Pheromones evaporate and diffuse away over time. So the remaining strength of the trail that was laid by one ant is a function of the original strength and the time since the trail was laid. If several ants use the same path the pheromone trail at this path will consist of the superimposed trails of these ants. The strength of the pheromones which is sensed by the

ants is the composite trail off all these trails.
The experiments for the investigation of the route optimization are visualized in Figure 20. It shows ants of an ant colony moving between the nest and a food source at three different moments. The ants can choose between two paths. The path on the left is longer then the path on the right. When the first ants start to search for food no pheromone trail exists. The ants make a random decision with a probability of 0.5 for turning left or right at the intersection near to the nest. So approximately 50 percent of the ants go along the shorter path and the other 50 percent go along the longer path.
After the first ants started a pheromone trail exists. But the trails on both paths are equally strong because both paths were used by a similar number of ants. So the probabilities at the intersection near to the nest remain approximately 0.5 for both paths. This first phase of the route finding process is illustrated in picture 1 of Figure 20.

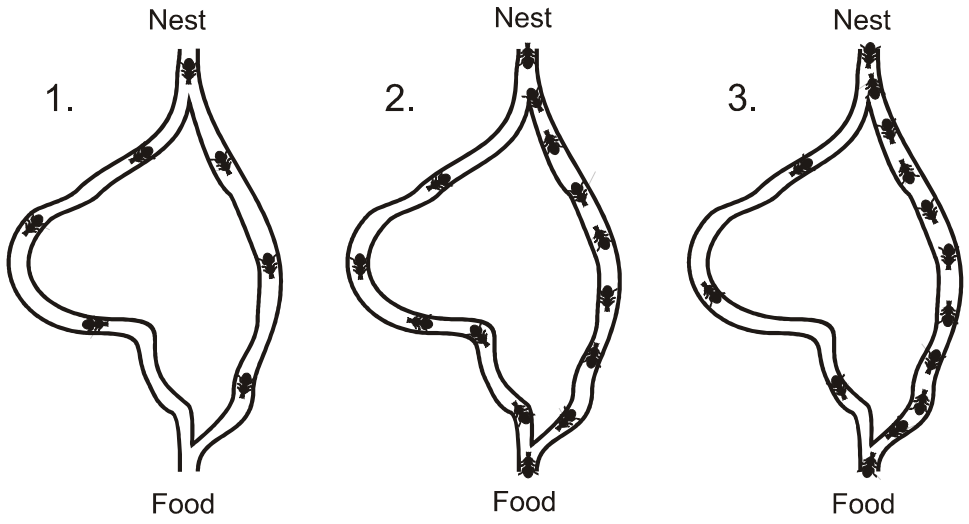

Figure 20: Pheromone following of ants

Because the right path is shorter the ants that move along this path reach the food source first. When the first ants that used the shorter path to the food source go back to the nest they follow the shorter path with a much higher probability than the longer path because on the shorter path exists a pheromone trail but not on the longer path. This is because no ant has arrived along the longer path yet. Note that the probability for deciding for one of the two paths at the intersection near to the nest remains 0.5 because no ant has arrived at the nest yet.
When the first ants of the longer path arrive at the food source, the probability to choose the longer path to go back to the nest increases. That is because now a pheromone trail exists also at this path. But the probability to go back along the shorter path is still much higher because this pheromone trail consists of many superimposed trails and is much stronger. This means that the large part of the ants goes back along the shorter path, but nevertheless some ants will go back along the longer path. This is shown in picture 2 of Figure 20.
The ants that go back along the shorter path arrive earlier at the nest than the ants that go along the longer path. So at the moment when the first ants arrive back at the nest the pheromone trail of the shorter path at the intersection near to the nest is strengthened. From

now on the probability for the ants which want to go to the food source is higher for the shorter path. This means that more ants decide to go along the shorter path and less to go along the longer path. In that way the pheromone trail of the shorter path and so the probability to go along the shorter path is even more increased. Picture 3 of Figure 20 shows this situation. There are only very few ants which use the longer path but nearly all ants use the shorter path. So the ants have optimized the route to the food source after a certain time.

5.3 Ant Based Control algorithm

The ABC-algorithm is derived from the pheromone following of ants that optimize the route to a food source. The natural stigmergy pheromone is replaced by artificial stigmergy which can be modelled by computers. Artificial stigmergy is defined in [Dorigo 1999] as the "indirect communication mediated by numeric modifications of environmental states which are only locally accessible by the communicating agents". For the ABC-algorithm the artificial stigmergy is realized by special data structures, called routing tables.

5.3.1 Routing tables

A routing table is a conventional table with rows and columns and belongs to exactly one node of the network. At every node of the network there is a locally stored routing table. For every neighboured node that can be reached from a node the routing table of the node contains a column. And for every node of the network except the node itself the routing table contains a row. The entries in the table are numeric values between 0 and 1 which represent the probability to reach other nodes represented by the rows along the neighboured nodes in shortest time. The probabilities in each row sum up to 1. A special notation is used to give all entries a unique name: $P_{destinationNode,\ neighbouredNode}$. The first variable, *destinationNode*, defines the row in the routing table. And the second variable, *neighbouredNode*, defines the column. The probability $P_{destinationNode,\ neighbouredNode}$ therefore is the probability to reach the destination node in shortest time by going along the neighboured node from the current node.
Let's consider an example of a routing table now. Table 2 is the routing table of node 5 from the network shown in Figure 21. Because the neighboured nodes of node 5 are the nodes 2 and 4 there exists a column for each of them. For every other node except node 5 there is a row in the routing table. The probability to reach node 2 in shortest time by directly going to node 2 ($P_{node2,\ node2}$) is very high, namely 0.95. It is also possible to go to node 2 along node 4. But the probability to go there in shortest time ($P_{node2,\ node4}$) is very small, only 0.05. The probability to go from node 5 to node 3 in shortest time along node 4 ($P_{node3,\ node4}$) is 0.73 and along node 2 ($P_{node3,\ node2}$) 0.27. The probabilities to go to node 1 in shortest time do not deviate so strongly from each other. $P_{node1,\ node2}$ is 0.59 and $P_{node1,\ node4}$ is 0.41. That is because the route to node 1 along node 2 is not much shorter than the route to node 1 along node 4.

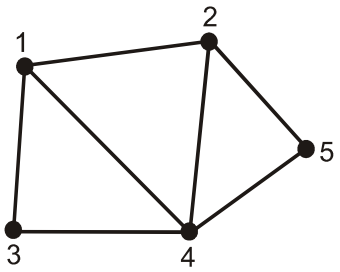

Figure 21: A simple traffic network

Table 2: Routing table of node 5 from the network of Figure 21

Destination node ↓ | Next node →	node 2	node 4
node 1	0.59	0.41
node 2	0.95	0.05
node 3	0.27	0.73
node 4	0.06	0.94

The probabilities in the routing tables can be compared with the strength of the pheromone trail. The higher the probability the stronger the pheromone trail. Note that the routing tables only contain local information on the best routes and no global information.
To compute the shortest routes in time for vehicles from any node in the network to any other node the information of all routing tables on the path from the source node to the destination node is needed. The next step from a node towards the destination node is always determined by using the routing table of the node. The next node is the node with highest probability in the row that represents the destination. By that vehicles are always routed to their destination along the path with highest probability. And since this path represents the strongest pheromone trail to the destination it should be the shortest known path in time.

5.3.2 Intelligent agents

What has not been considered so far is how the probabilities in the routing tables are computed. In the nature ants lay pheromone and so they produce pheromone trails between the nest and a food source. On a computer the pheromone has been replaced by artificial stigmergy, the probabilities in the routing tables. To compute and update the probabilities, intelligent agents are introduced to replace the ants. There exist two kinds of agents, the forward agents and the backward agents. All forward agents have the same structure and all backward agents have the same structure. The agents move inside the network by hopping at every time step from a node to the next node along the existing links. Note that the agents can move much faster than the vehicles could do in the real traffic network. They communicate with each other in an indirect way by concurrently reading and writing the routing tables on their way. The forward and backward agents can be classified as reflex agents with internal state according to [Russell 1995]. They receive percepts from the environment and they can do certain actions. Condition-action rules are used to define what

action should be chosen under which situation. The current situation is defined by the percept and the stored internal state. A PAGE description of both agent types is given in Table 3. The condition-action rules of the agents are defined in Table 4.

Table 3: PAGE description of forward and backward agents

Agent Type	Percepts	Actions	Goals	Environment
Forward agent	−ID of current node −IDs of the neighboured nodes and links that lead to them −routing table of current node	−update memory −remove cycle from memory −determine and go to next node −transform to backward agent	go to destination node, store the route and the travel times	network consisting of nodes and directed, weighted links connecting the nodes, only local information available at every node (routing tables)
Backward agent	−ID of current node −IDs of the neighboured nodes and links that lead to them	−update routing table −go to next node −kill the agent	go back to the source node along the stored route of the forward agent and update the routing tables	see forward agent

Table 4: Condition-action rules of the agents

Agent Type	Condition-action rules
Forward agent	IF current node already exists in memory THEN remove cycle from memory IF current node ≠ destination node THEN update memory AND determine and go to next node IF current node = destination node THEN update memory AND transform to backward agent
Backward agent	IF current node ≠ source node THEN update routing table AND go to next node IF current node = source node THEN update routing table AND kill the agent

The internal state of the agents is a kind of memory which stores a list of (k, t_k)-pairs. Every such pair represents a node that has been visited by the forward agent. k is the identifier of the visited node and t_k is the time it will take a car to travel the link from the last visited node to this node under the current traffic situation. The time t_k is taken from the timetable of the Routing system. So t_k must be understood as a virtual delay that is assigned to the agents to simulate that the agents go along the streets like the vehicles. But the agents indeed move inside the computer system only and can move much faster than the vehicles can do.

5.3.3 The algorithm

Using the condition-action rules from Table 4 and the PAGE description of the agents from Table 3 the ABC-algorithm can be explained now:

- At regular time intervals from every node *s* of the network a forward agent F_{sd} is launched with a random destination node *d*. The task of the forward agent is to discover a feasible, low-cost path to the destination.
- At every visited node *k* on the way to the destination node a forward agent does the following:
  - The forward agent checks its memory whether node *k* has already been visited by the agent. If this is the case then a cycle in the agent's path exists. This cycle is deleted from the memory.
  - The forward agent updates its memory by adding a new (k, t_k)-pair to the memory.
  - If node *k* is not the destination node then the forward agent determines the next node to go to by using the probabilities in the row of the routing table of node *k* which represents the destination node. A random number between 0 and 1 is generated for every link to a neighboured node according to the magnitude of the probabilities. The node where the agent just came from is filtered out to avoid that the agent directly goes back to that node. Also links that are disabled are filtered out. With the generated probabilities the next link is randomly selected. The forward agent goes to the next node along that link.
  - If node *k* is the destination node then the forward agent F_{sd} transforms to a backward agent B_{ds}. The backward agent inherits the memory from the forward agent. The task of the backward agent is to go back to the source node *s* along the same path as the forward agent but in the opposite direction and to update the routing tables on this path.
- At every visited node *k* on the way back to the source node a backward agent does the following:
  - The backward agent updates the routing table of node *k* by using the travel times stored in its memory. The update process is explained in the next section, section 5.3.4.
  - If node *k* is not the source node then the backward agent uses its memory to determine the next link of the path back to the source node. The backward agent goes to the next node along that link.
  - If node *k* is the source node then the backward agent is killed.

Let's summarize the indirect communication of the intelligent agents via artificial stigmergy:
The forward agents use the routing tables (artificial stigmergy) of the nodes to find their way to the destination node. They only have reading access to the routing tables and can not change the probabilities. The backward agents use their memory to find the way back to the source node. On their way back they update the routing tables by changing some of the probabilities. By that the probabilities in the routing tables change very fast, possibly several times a second. The next forward agents that visit the nodes will find other values in the routing tables and therefore other conditions under which they react. By this indirect communication between the forward agents and the backward agents an emergent behaviour of the quite simple agents arises like in ant colonies what leads to the ability to optimize routes.

5.3.4 Updating the routing tables

At every visited node *k* on the way back to the source node, a backward agent updates some of the probabilities in the routing table of node *k* by using the travel information in its memory which was collected by the forward agent.
For the following let's call the next node after node *k* on the forward agent's path to the destination node *k+1*, the second next node *k+2* and so on. The destination node is node *k+n*. The backward agent's memory contains the virtual travel times which vehicles would need to go from every node to the respective next node on the path to the destination node. So the backward agent knows among other times the time t_{k+1} to go from node *k* to node *k+1*, the time t_{k+2} to go from node *k+1* to node *k+2* and so on. This is displayed in Figure 22. Knowing these times the backward agent can compute the time to reach node *k+2* from node *k*, the time to reach node *k+3* from node *k* and so on by simply adding the times of the links between both nodes.

Figure 22: Part of a forward agent's path

Now let's consider what probabilities a backward agent updates in the routing table of node *k* on its way back to the source node. Table 5 shows the probabilities that are updated and the times which are used for the update. To summarize the content of Table 5, not only the probability to reach the destination node *k+n* from node *k* along node *k+1*, but also the probabilities to reach every other node on the sub path to the destination node along node *k+1* are updated (nodes *k+1*, *k+2*,…).

Table 5: The probabilities that are updated by the backward agent at node *k*

Probability to update	Used travel time
$P_{k+1,k+1}$	t_{k+1}
$P_{k+2,k+1}$	$t_{k+1}+t_{k+2}$
$P_{k+3,k+1}$	$t_{k+1}+t_{k+2}+t_{k+3}$
…	
$P_{k+n,k+1}$	$t_{k+1}+t_{k+2}+t_{k+3}+\ldots+t_{k+n}$

The probabilities $P_{destination,k+1}$, where *destination* is a place holder for the nodes *k+1*, *k+2*,…, *k+n*, are updated by using the following formula:

$$P_{destination,k+1} = \frac{P^{old}_{destination,k+1} + \Delta P}{1 + \Delta P} \qquad (1)$$

- $P_{destination,k+1}$ is the new probability
- $P^{old}_{destination,k+1}$ is the old probability in the routing table

- ΔP is the probability increase

Note that the formula (1) always increases the probability $P_{destination,\,k\,+\,1}$, what means that every update is a positive feedback. This is like with the pheromone of the ants where every dropped pheromone increases the strength of a pheromone trail but not weakens it.

ΔP is computed with the following formula:

$$\Delta P = \frac{A}{t} + B \qquad (2)$$

with
- A, B constants; convenient values are: A=0.8, B=0.01
- t the trip-time of the forward agent from node k to node *destination*

Formula (2) ensures that the probability increase ΔP is inversely proportional to the travel time of the forward agent: The higher the time t, the lower the probability increase ΔP. And the lower the time t, the higher the probability increase ΔP. By that good paths receive a strong update while bad paths receive a low update only.
For a given value of ΔP, the absolute and relative increase of $P_{destination,\,k\,+\,1}$ is much larger for small values of $P^{old}_{destination,k+1}$ than for large values of $P^{old}_{destination,k+1}$. By that weighted change low probabilities go up very fast to adapt to the new traffic situation whereas probabilities which are already high are increased only a little bit.

Whenever a probability in the routing table is increased by formula (1) the other entries in the same row must be decreased. This is necessary to ensure that the sum of the probabilities in each row remains 1. The following formula is used to normalize the values:

$$P_{destination,\,i} = \frac{P^{old}_{destination,i}}{1 + \Delta P} \quad \text{for all neighboured nodes } i \text{ of node } k \text{ with } i \neq k+1 \qquad (3)$$

This means that probabilities can only decrease if another probability in the same row is increased. A probability can approach zero if other probabilities in the same row of the routing table are increased much more often or much stronger. This would mean that hardly ever a forward agent takes the route associated with this low probability, what means that this probability receives nearly no further updates. But indeed this route can become the fastest route in time under changed traffic conditions. To be able to detect this it must be ensured that no probability in the routing table reaches zero. This can be done by introducing an exploration probability as a minimum value for each probability. An example could be 0.05 divided by the number of neighboured nodes. Whenever a probability is smaller than the exploration probability this probability is set to the exploration probability. After setting this minimum the probabilities per destination are normalized again. By that it is ensured that every link is used by forward agents from time to time and so new routes are explored regularly.

The value of a probability in a routing table finally depends on two facts: the trip times experienced by the forward agents that use the route associated with the probability and the frequency of the updates (the number of agents that use the route). For example if the trip times are very low, the probability increase ΔP is very high and so the probability is strongly increased with every update. So the probability will be very high even if the frequency of the update is low. But on the other hand also high trip times, which mean a low probability increase ΔP, can lead to a high probability if the frequency of the updates is high. In that way high frequent updates with low travel times lead to the highest probabilities.

What has to be mentioned is that the formulas (1) and (2) to update the probabilities are very simple. They only take the old probability and the experienced trip time to compute the new probability. The adaptation to suddenly occurring new traffic situations is very slow. There exist much better but more complicated approaches that update the probabilities faster when the travel times dramatically change. Such approaches use heuristics and statistical approaches that consider for example also the last n trip times that have been experienced (see for example [Tatomir 2002]). Nevertheless for this thesis we have decided to overtake the formulas (1) and (2) that were used in [Kroon 2002], because the formulas are not of importance for the mode of operation of the Hierarchical routing system. But to achieve a better performance of the ABC-algorithm these formulas are the key for an improvement.

5.4 Hybrid system

In this section, an important part of this thesis, it is considered how the ABC-algorithm can be adapted to be used by the Hierarchical routing system.
The Hierarchical routing system consists of a Global route finding system and several Sector route finding systems (see Figure 9 in section 3.2). The Global route finding system is responsible for routing vehicles on the abstract level of the traffic network. To the abstract level belong all slip roads/departures, motorway intersections and motorway links that connect them. And a Sector route finding system is responsible for routing vehicles within the corresponding sector. All these Route finding systems can operate independently to calculate routes within the area they are responsible for. They all use their own timetable which stores the time estimates for the links of this area. The Route finding systems have own routing tables, exactly one routing table for each node of the network they are responsible for. Because nodes can belong to several sectors and therefore to several Routing systems there can be several routing tables for one node of the network.
To illustrate this an example is given in Figure 23. The shown network consists of four sectors, where three of them contain a city network. Therefore the Hierarchical routing system consists of a Global routing system and three Sector routing systems. Let's consider the labelled nodes:

- node A:
  For the node labelled with A only one routing table exists in the Hierarchical routing system because this node is an intersection and belongs to the Sector routing system of sector 3 only. The routing table is stored in the Sector route finding system of sector 3.

- node B.
  Because node B is a node of the motorway and belongs to the abstract level and sector 2, there exist two routing tables for this node. One routing table is stored in the Global route finding system and the other in the Sector route finding system of sector 2.

- node C:
  For the slip road/departure labelled with C three routing tables exist, one in the Global route finding system, one in the Sector route finding system of sector 2 and one in the Sector route finding system of sector 4.

- node D:
  For the motorway intersection D exist four routing tables, one in the Global route finding system, one in the Sector route finding system of sector 2, one in the Sector route finding system of sector 3 and one in the Sector route finding system of sector 4.

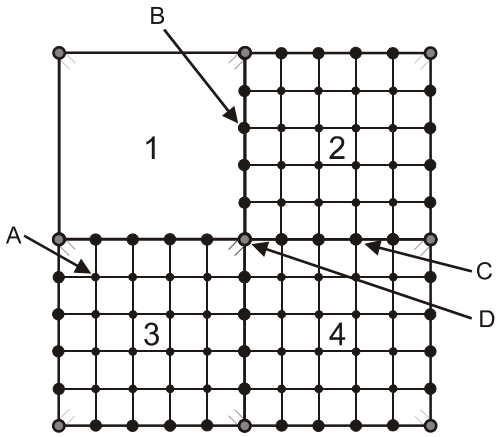

Figure 23: Traffic network with four sectors

Every routing table of a Route finding system contains rows for all other nodes of the network for which the Routing system is responsible for. This means that the routing tables of a Sector route finding system contain a row for every other node of the sector including the nodes of the surrounding motorway ring. And the routing tables of the Global route finding system contain a row for every other node of the abstract level. These are all slip roads/departures and motorway intersections of the network.
Every Route finding system has its own agents that explore the Routing system's network to update the routing tables. The agents move inside this area only and are not able to leave it. The possible destinations of forward agents are the nodes that belong to the same Routing system. And therefore the backward agents only update the routing tables of the Routing system where the agents also belong to.

Whenever a car should be routed from a node within a sector to another node within the same sector, the car is routed by the corresponding Sector route finding system by using the ABC-algorithm like explained in section 5.3. This means that the route is computed by using the resulting probabilities in the routing tables of each node on the path to the

destination node. The next node from any node on the path to the destination node is always the node with highest probability in the row which represents the destination. In the same way the Global route finding system computes routes for vehicles from one node of the motorway to another node of the motorway. So in both cases the single Route finding systems work in the same way as the Route finding system that was introduced in [Kroon 2002].
But if a route is needed from a node within one sector to a destination in another sector several Route finding systems of the Hierarchical routing system must cooperate to compute the route like mentioned in section 3.2.3. One way to compute the route is to combine two different techniques in one system. These are the ABC-algorithm as such, which updates the probabilities in the routing tables and routes vehicles by using the probabilities in the routing tables only, and additional computation to choose the motorway intersection of the source sector to leave the sector. This approach is followed and developed in the next sections and is called Hybrid system because of the combination of two contrasting techniques.

Now let's consider how the route from a node within a sector to a node within another sector is computed by the Hybrid system. The first step in the route computation process is the computation of the most suitable motorway intersection of the source sector. This is the motorway intersection for which the time to reach it plus the time to reach the destination sector along the motorway is minimal. For this four different routes need to be considered. These are the routes from the source node to one of the motorway intersections of the source sector along the path with highest probability and from there on the motorway to the destination sector along the path with highest probability. The routes which do not leave the source sector along the motorway intersection but which continue on a motorway of the source sector are filtered out. This is because the most suitable motorway intersection must be a motorway intersection along which the source sector is left. Now the travel times of the remaining routes are compared with each other and the motorway intersection along which the shortest route of the remaining routes goes becomes the most suitable motorway intersection. That was the part of the route computation process where different routes are computed and compared. For all further steps after the most suitable motorway intersection has been computed the route computation is done by using the probabilities in the routing tables only, which are updated by the ABC-algorithm. So the vehicle is routed to the most suitable motorway intersection by the Sector route finding system of the source sector first. If this motorway intersection is reached the vehicle will be routed along the motorway to the destination sector by the Global route finding system. And when the destination sector is reached the part of the route to the destination node is computed by the Sector route finding system of the destination sector.

To facilitate this course of cooperation some extensions and refinements must be made to the Route finding systems. The major changes are:

- introduce a virtual node for each sector that contains a city network and expand the routing tables of the abstract level with rows for the virtual nodes
- add time columns to every routing table of the Sector route finding systems and to the routing tables of the motorway intersections of the Global route finding system

Furthermore some special adjustments must be made to the Sector route finding systems to deal with special cases and to ensure a good performance.
These changes are described in the next sections.

5.4.1 Refinements

5.4.1.1 Virtual nodes

For every sector that contains a city network a virtual node is introduced. Virtual nodes are used on the abstract level of the network. A virtual node is the union of all nodes that belong to the same sector. So a virtual node can be understood as abstraction from all the nodes of the sector. It represents one sector of the detailed level of the network (see Figure 5) which contains a city network. Figure 24 illustrates that principle of a virtual node.
Whenever a car is routed from an intersection of one sector to an intersection of a distant sector the car must be routed along the motorway to the destination sector. And this destination sector is represented by its virtual node. So in that case cars are routed to the virtual node by the Global route finding system. When this virtual node is reached, what means that the sector is reached, the Sector route finding system of that sector is used to compute the last part of the route.

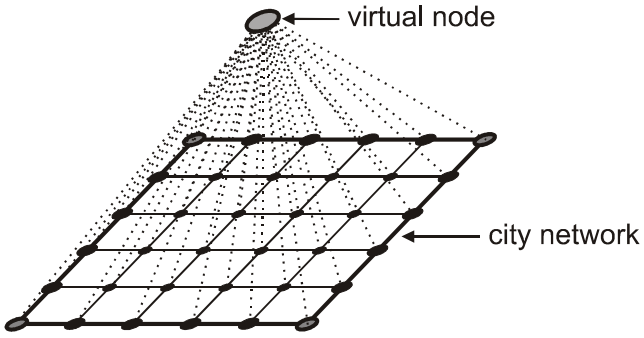

Figure 24: Virtual node of a city network

5.4.1.1.1 Expanding the routing tables

To realize the routing to a virtual node the routing tables of the Global route finding system (these are the routing tables for the abstract level) must be expanded. These routing tables must have now also entries for the virtual nodes. So for each virtual node there is now a row in the routing table that contains the probabilities for every neighboured node to reach the virtual node along the neighboured node. This means that the abstract level now consists not only of all slip roads/departures and motorway intersections but also of the virtual nodes. And this is illustrated by Figure 25 which shows the abstract level of the traffic network example that was introduced in Figure 4.

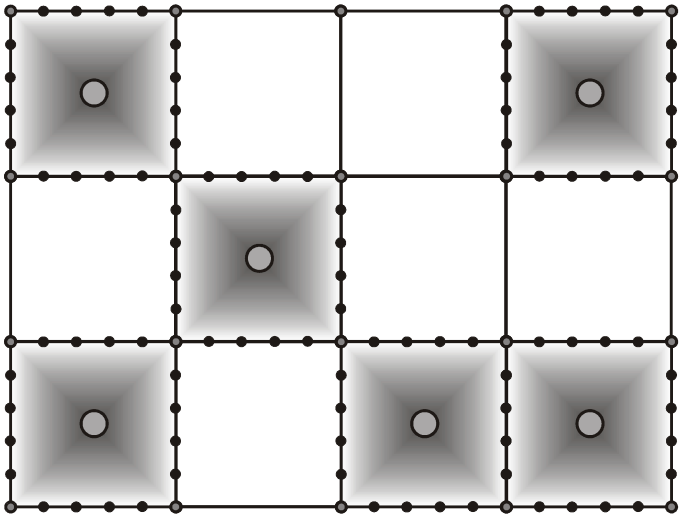

Figure 25: Abstract level of the traffic network from Figure 4 with virtual nodes

Let's consider an example of a routing table with virtual nodes. Table 6 shows the routing table of the Global route finding system of node 46 from the network shown in Figure 26. The network has 68 nodes and 6 sectors where the sectors 2, 3, 4 and 6 contain a city network. Because node 46 is a slip road/departure it has only two neighboured nodes. These are the nodes 42 and 50. So there are only two columns in the routing table as the only possible next nodes. For each node of the abstract level except the node 46 itself there is a row in the routing table that contains the probability for each of the neighboured nodes. The sum of the probabilities in each row is 1. A high value means a high probability to reach the destination in shortest time along the neighboured node.
Besides the rows for all nodes of the abstract level there are now also rows for sectors that contain a city network. For each of these sectors except the sectors where the node itself belongs to (this is sector 4) exists a row in the routing table. So for example the probability to go to sector 2 along node 42 is 0.81 and 0.19 along node 50, which indeed is quite obvious under normal traffic conditions when looking at Figure 26.

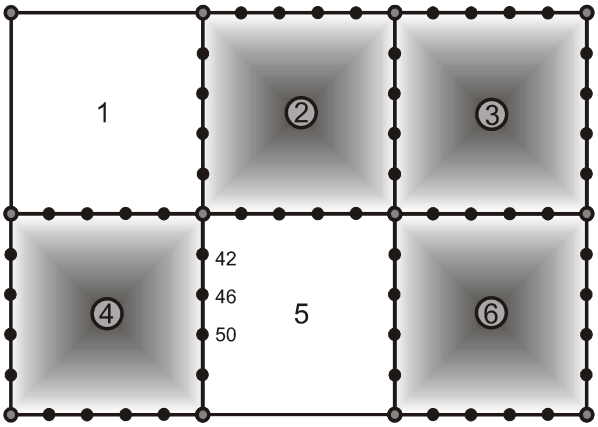

Figure 26: Abstract level of a traffic network

Table 6: Routing table of the Global route finding system of node 46

Destination ↓ | Next node →	node 42	node 50
node 1	0.68	0.32
node 2	0.71	0.29
...		
node 42	0.95	0.05
…		
node 50	0.06	0.94
…		
node 68	0.34	0.66
sector 2	0.81	0.19
sector 3	0.72	0.28
sector 6	0.54	0.46

5.4.1.1.2 Updating the probabilities

Now it must be considered how to compute the probabilities for the virtual nodes in the routing tables of the Global route finding system. For the nodes forward and backward agents are used to update the entries in the routing tables like explained in section 5.3. So at regular intervals a forward agent is launched from every node of the abstract level with a random destination. The destination node can be any other node of the abstract level. When the destination is reached the forward agent transforms to a backward agent which inherits the memory of the forward agent that contains the recorded route information. This backward agent now goes along the recorded route but in the other direction and updates the routing tables of the nodes with the data in the memory.
Fortunately the same forward and backward agents can be used to update also the entries for the virtual nodes in the routing tables. The backward agents already have sufficient information in their memory to update the sectors. Only the update behaviour of the backward agents needs to be extended. The idea is that whenever a forward agent reaches a new sector on its way to the destination node, the probabilities for the virtual node of the sector in the routing tables of all nodes before the sector are updated by the corresponding backward agent with the travel times in the agent's memory.
Figure 27 shows the update procedure for the backward agents of the Global route finding system in pseudo code.

```
procedure UpdateRoutingTable(currentNode, memory)
 inputs:  currentNode, the node where the backward agent is
          memory, the memory of the backward agent

 nextNode:=the successor node of currentNode taken from memory
 destinationNode:=the destination node of the route taken from memory
 subpathNode:=currentNode
 currentNodeSectors:=the Set containing all sectors the currentNode belongs to
 updatedSectors:=an empty Set
 travelTime:=0
 repeat
   subpathNode:=the successor node of suppathNode taken from memory
   subpathNodeSectors:=the Set containing all sectors the subpathNode belongs to
   travelTime:=travelTime + time to reach subpathNode from last node taken from memory
   update the probability P_subpathNode, nextNode in the routing table of currentNode with travelTime
   for each Sector in (subpathNodeSectors \ currentNodeSectors) \ updatedSectors that contains
   a city network do
     update the probability P_Sector, nextNode in the routing table of currentNode with travelTime
     add Sector to updatedSectors
   end
 until subpathNode = destinationNode
```

Figure 27: Update procedure for the backward agents of the Global route finding system

At every node on the way back to the source node this procedure is executed by the backward agent. The procedure updates the probabilities for nodes as well as the probabilities for virtual nodes in the routing table of the node where the backward agent actually is.

Not only the path to the destination node but also the sub paths are updated. Therefore the repeat loop switches in every run to the next node on the way to the destination until the destination node is reached. In every run of the loop the computed *travelTime* is used to update the probability $P_{subpathNode,\ nextNode}$ in the routing table what means that this probability is incremented. This is the probability to reach the *subpathNode* by going along the *nextNode* from the *currentNode*. All other probabilities in the same row of the routing table must be decremented so that the sum of the row remains 1 (see section 5.3.4). Also the probability $P_{Sector,\ nextNode}$ for every newly entered sector is updated with the computed *travelTime* what means that this value is incremented and that again all other probabilities in the same row are decremented so that the sum remains 1.

To compute the newly entered sectors mathematical sets and the mathematical set operation "\" (difference) is used. Newly entered sectors are the sectors in the set *subpathNode* that are not contained in the set *currentNodeSectors* and that have not been updated yet. And these are the sectors that are contained in the set
(*subpathNodeSectors* \ *currentNodeSectors*) \ *updatedSectors*.

Let's consider an example to illustrate this update procedure. Figure 28 shows the last part of the route of a forward agent that belongs to the Global route finding system. Node 28 is the destination of the route. When the forward agent arrives at node 28 it transforms to a backward agent. The backward agent goes along the route in the opposite direction along the nodes 27, 26, 25 and so on. At every node the procedure UpdateRoutingTable is executed and the node's routing table is updated.

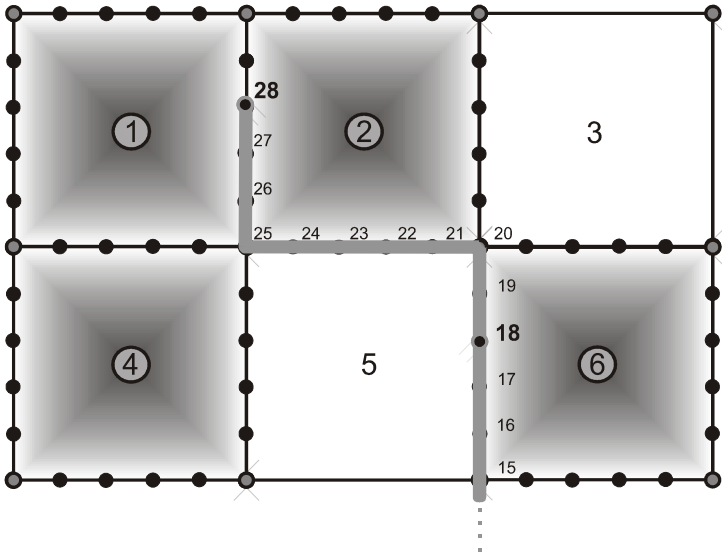

Figure 28: A route of a forward agent of the Global route finding system

Let's regard the execution of the update procedure when the backward agent already has passed the nodes 27 till 19 and is now at node 18. So the parameter *currentNode* is set to node 18:

- In the first run of the repeat loop the *subpathNode* is set to node 19. The probability $P_{node19,\ node19}$ is updated. This is the probability to reach node 19 by going to node 19. Because no new sector was entered by going from *currentNode* to *subpathNode* no sector probability is updated.
- In the second run of the loop the *subpathNode* is set to node 20. Now the probability $P_{node20,\ node19}$ is updated. That is the probability to reach node 20 by going to node 19. Two new sectors have been entered. These are the sectors 2 and 3. But only sector 2 contains a city network. So the probability $P_{Sector2,\ node19}$ is updated. Sector 2 is now added to the set *updatedSectors*.
- In the third run of the loop the *subpathNode* is set to node 21. The probability $P_{node21,\ node19}$ is updated. No new sector was entered and therefore no other probability can be updated.
- The same continues in the next three runs of the loop. Only the probability $P_{subpathNode,\ nextNode}$ is updated in each run.
- In the loop when *subpathNode* is set to node 25 the sectors 1 and 4 are updated because

they newly have been entered. This is the last time during the execution of the procedure UpdateRoutingTable that probabilities for sectors are updated.

- In the next and last three loops only the probability $P_{subpathNode,\ nextNode}$ is updated.

To sum up, the rows for the nodes 19 till 28 and the rows for the sectors 1, 2 and 4 are updated in the routing table of node 18 during the execution of the procedure UpdateRoutingTable.

5.4.1.2 Time columns

To compute the most suitable motorway intersection like explained in section 5.4 some additional information must be added to the routing tables. For every node of a sector (these are the intersections, slip roads/departures and motorway intersections of the sector) the time to reach each of the four motorway intersections of the sector must be available. And for every motorway intersection the time to reach every other destination of the abstract level is needed. To realize this all routing tables of the Sector route finding systems and the routing tables of the Global route finding system for the motorway intersections are extended with an additional column, called the time column. The time values in the time column shall represent the estimated time in seconds for the vehicles to reach the destination that is represented by the row. Because the vehicles are always routed along the nodes with highest probability to their destination the time values shall represent the time for the route to the destination along the nodes with highest probability.
In the routing tables of the Sector route finding systems there is only a time value in the rows of the time column that represent a motorway intersection. For normal intersections and slip roads/departures these are always four rows for the defined example network because each sector has got four motorway intersections. In the routing tables of the motorway intersections there are three rows which contain a time value because the motorway intersection itself is not contained as a destination. Of course the time to reach this motorway intersection itself is 0 seconds.
To illustrate the design of a routing table of an intersection an example is given.
Table 7 shows the routing table of node 23 of the Sector routing system which is responsible for the city network shown in Figure 29.

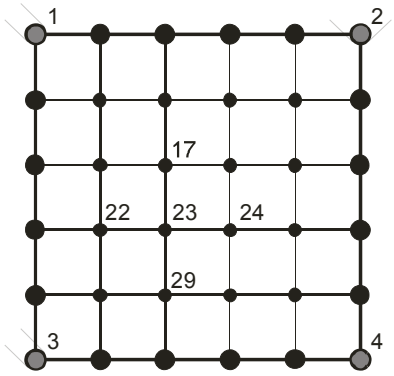

Figure 29: Example of a sector network

Table 7: Routing table of node 23 of the Sector route finding system

Destination ↓ | Next node →	node 17	node 22	node 24	node 29	time
node 1 (motorway intersection)	0.39	0.52	0.05	0.04	26 s
node 2 (motorway intersection)	0.42	0.07	0.46	0.05	35 s
node 3 (motorway intersection)	0.07	0.38	0.04	0.51	21 s
node 4 (motorway intersection)	0.11	0.09	0.38	0.42	30 s
node 5	0.62	0.29	0.05	0.04	-
…					
node N	0.04	0.04	0.54	0.38	-

The time column is filled with four time values, one for each motorway intersection of the sector. For node 23 the closest motorway intersection in time is node 3 (21 seconds). And the node with highest probability to go there is node 29.

In the routing tables of the Global route finding system for the motorway intersections there is a time value in each row of the time column. But the motorway intersection itself and the virtual nodes of the sectors to which the motorway intersection belongs are not contained in the routing table. That is because these destinations are already reached when a vehicle is at this motorway intersection. So the time to reach these destinations is 0 seconds.
Let's consider an example of a routing table of the Global route finding system for a motorway intersection. Table 8 is the routing table of node 30 from the network displayed in Figure 30. The network has 68 nodes and consists of six sectors where the sectors 2, 3, 4 and 6 contain a city network. But the only virtual nodes in the routing table are the virtual node for sector 3 and the virtual node for sector 6, because node 30 belongs to the sectors 2 and 4. Every row of the time column is filled with a time value. For example the approximate time to reach sector 3 is 29 seconds by going along node 31.

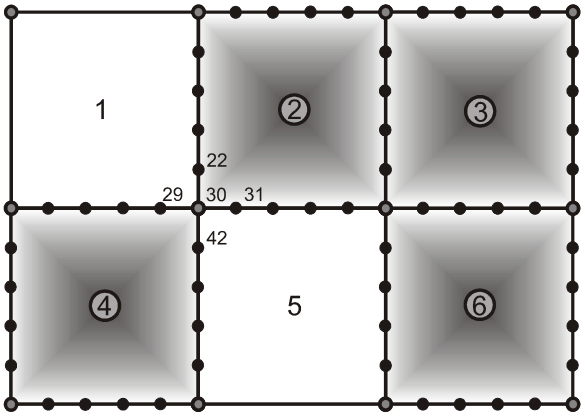

Figure 30: Abstract level of a traffic network

Table 8: Routing table of node 30 of the Global route finding system

Destination ↓ | Next node →	node 22	node 29	node 31	node 42	time
node 1	0.45	0.43	0.05	0.07	55 s
node 2	0.61	0.16	0.19	0.04	51 s
...					
node 22	0.92	0.03	0.02	0.03	6 s
…					
node 29	0.02	0.93	0.02	0.03	5 s
node 31	0.03	0.02	0.92	0.03	6 s
…					
node 42	0.02	0.03	0.02	0.93	7 s
…					
node 68	0.05	0.10	0.39	0.46	91 s
sector 3	0.18	0.08	0.69	0.05	29 s
sector 6	0.06	0.03	0.70	0.21	29 s

Now the question is how to fill the time columns. Therefore two different possibilities to compute the time values are considered in the next two sections.

5.4.1.2.1 Observation window

One way to compute the time values in the routing tables is the introduction of observation windows to nodes whose routing tables contain time values. For every destination in the routing table with a time value an observation window is necessary. This principle is displayed in Figure 31 which shows a routing table of an intersection and the accompanying observation windows. This routing table belongs to a Sector route finding system and contains like every intersection four time values. For each of them an observation window exists.

Destination ↓ | Next node →	node 17	node 22	node 24	node 29	time	
node 1 (motorway intersection)	0.39	0.52	0.05	0.04	26 s	→ observation window for node 1
node 2 (motorway intersection)	0.42	0.07	0.46	0.05	35 s	→ observation window for node 2
node 3 (motorway intersection)	0.07	0.38	0.04	0.51	21 s	→ observation window for node 3
node 4 (motorway intersection)	0.11	0.09	0.38	0.42	30 s	→ observation window for node 4
node 5	0.62	0.29	0.05	0.04	-	
…						
node N	0.04	0.04	0.54	0.38	-	

Figure 31: Routing table of an intersection with accompanying observation windows

Observation windows can be understood as a list which stores the last travel times. An observation window at node *k* stores the last *w* travel times the last *w* forward agents needed for their route from node *k* to the destination. It can be implemented like a FIFO list (first in first out) where the elements are ordered according to their age. The element that is the oldest one is at position 1 in the list. And the newest element is stored at position *w*, the last position in the list. Figure 32 displays an observation window with *w* elements.

Whenever a new element should be added, the element at position 1 is removed and the new element is added at the end of the list. So the former element at position 2 is now at the first position and is now the oldest element in the list. The number of elements in the list always remains constantly at the value of *w*.

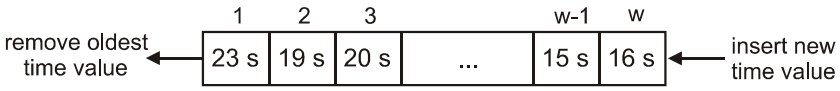

Figure 32: Observation window with *w* elements

The travel times, which are to be stored in the observation window, are available in the memory of the backward agents which come from the nodes. So at every node on the way back to the source node a backward agent updates not only the probabilities in the routing table of the node but also stores the route times in the corresponding observation windows of the node. This is done with the travel time from the current node (the node where the backward agent is) to every node on the path to the destination node including the destination node for which a time value in the routing table of the current node is needed.
With the times in the observation window the time value can be computed at every time when the observation window is updated. The easiest way for the computation of a time value is to calculate the average value of all times in the observation window. A more accurate way would be the computation of a weighted average. Newer route times should have a stronger influence on the average time than older route times. So newer route times are weighted with a higher factor. This approach is justified through the fact that new route times reflect much better the actual situation in the traffic network than older ones. Another way would be to consider the first derivative of the travel times. A high value of the first derivative means a strong change of the travel times. In that case the newer route times should receive a higher weight for the computation of the average to ensure a quick adaptation.

For the observation window approach it is very important to remark that the majority of the times in the observation window are not the times for going along the nodes with highest probability. The reason is that forward agents are not routed along the nodes with highest probabilities. Forward agents always choose the next node randomly according to the magnitude of the probabilities. So in most of the cases a forward agent does not use the path along the nodes with highest probabilities. And that is why the average time computed with an observation window can not be understood as the route time to the destination along the nodes with highest probability. Rather it must be interpreted as an average value of all route times of the last *w* agents that went to the destination along the current node using very different routes. So what is not possible to realize with an observation window is the requirement that the time in the time column always represents the time for the route along the highest probability. Another problem is that if the route changes because another node on the route becomes the node with highest probability it will always need several update steps before the time values approach the new found route.

The exactness of the computed time values is strongly influenced by the size *w* of the observation window. If there are too few elements in the observation window elements that strongly deviate from the optimal route will falsify the average value a lot. And if there are

too many elements the average value will adopt to the actual situation in the traffic network with a high delay only. So some considerations must be made to determine a suitable value for *w*.

5.4.1.2.2 Time agents

The second way to compute the time values for the routing tables is the introduction of a third kind of agent, called time agent. The only usage of time agents is to update the time values in the routing tables. Like with forward and backward agents there are time agents that belong to the Global routing system and time agents that belong to a Sector routing system. A time agent moves within the network of the corresponding Routing system only.
At regular time intervals from every node whose routing table contains a time column a time agent is launched. The destination of the agent is one of the destinations in the routing table for which a time value is needed. A time agent moves to the destination along the nodes with highest probabilities, like vehicles are routed. At every node the time agent adds the time that was needed to reach the node from the previous node to the total route time. This time is taken from the timetable of the Routing system to which the agent belongs to. When the time agent reaches the destination, the total route time is sent back to the source node of the time agent. There this time is entered in the corresponding row of the time column by replacing the old time value. Note that the time measured by the time agent represents the time which a vehicle would need to go to the destination.
Like the forward and backward agents, which were introduced in section 5.3.2, the time agents can be classified as reflex agents with internal state. The PAGE description for the time agents is given in Table 9 and the condition-action rules are defined in Table 10.

Table 9: PAGE description of time agents

Agent Type	Percepts	Actions	Goals	Environment
Time agent	−ID of current node −IDs of the neighboured nodes and links that lead to them −routing table of current node	−update memory (add new travel time to total travel time) −go to next node −send the travel time to the source node and kill the agent	go to destination node along the path with highest probability and measure the travel time	network consisting of nodes and directed weighted links connecting the nodes, only local information available at every node (routing tables)

Table 10: Condition-action rules of the time agents

Agent Type	Condition-action rules
Time agent	IF current node ≠ destination node THEN update memory AND go to next node IF current node = destination node THEN update memory AND send the travel time to the source node and kill the agent

In the Global route finding system time agents are launched at regular time intervals from every motorway intersection. The destination of a time agent is one of the destinations in the routing table of the motorway intersection. So this can be a slip road/departure, motorway

intersection or sector that contains a city network. The time agent uses the path along the motorway with highest probability.
In the Sector route finding systems time agents are launched from all nodes. But the destinations are the motorway intersections of the sector only. So at each update interval a time agent is sent from every node to one of the motorway intersections.
To make sure that all time values are updated regularly after a certain time interval the destination of the time agent should not be chosen randomly but deterministically. This is possible by defining a fixed order for the destinations and an iterator that always points to the next destination in the order. At every update interval the destination of the launched time agent is the destination where the iterator points to. After launching the time agent the iterator must be updated to point to the next destination in the fixed order. If the end of the order is reached the iterator will have to be reset to the first element and the iteration starts again.

For the topicality of the time values in the time columns a short update interval is of great importance. But a shorter update interval means to send time agents more often and in that way more time agents exist in the network at the same time what requires more computation. So the update interval should be chosen deliberately.

5.4.1.2.3 Comparison

In this section both ways to compute the travel times for the routing tables, the observation window approach and the time agents, are compared with each other. Table 11 summarizes the comparison by considering the criterions exactness of the computed time values, adaptation speed to new traffic situations and the computational expense that is necessary.

Table 11: Comparison of the observation window approach with time agents

Criterion	Observation window	Time agent
exactness	computed time values strongly deviate from the travel time along the path with highest probability	computed time values exactly represent the travel time along the path with highest probability
adaptation speed	low adaptation speed, always several time steps are necessary for the adaptation because the travel time is computed as the average from all known travel times	very fast adaptation, the time agents always measure the actual travel time with a very low delay only
computational expense	low, existing forward and backward agents can be used, additional computation only necessary to compute the average time values, additional memory space necessary for the observation windows	higher, new type of agent is needed, more agents move within the network, more agents need access to the routing tables and timetable

Because the correctness of the time values in the time columns strongly influences the exactness and therefore the performance of the Hybrid system, the approach should be preferred which supplies the best time values. And as clearly can be seen in Table 11 this is the time agent approach. The only drawback of the time agents is the higher computational expense. But at this point in the optimization of the routes this expense really should be undertaken to guarantee the best possible performance for the Hybrid system.

It is also possible to combine both approaches. The time agent approach can be improved with observation windows. Instead of directly taking over the travel times experienced by the time agents, the last *w* travel times can be stored in an observation window. Then an average of the travel times can be computed to smooth out possible fluctuations of the travel times which can occur because of the randomness of the ABC-algorithm.

5.4.1.3 Further adjustments

Besides the refinements described in the last two sections some further adjustments for the Sector route finding systems are necessary. These adjustments are considered in this section.

5.4.1.3.1 Adjacent sectors

The first adjustment is the introduction of virtual nodes for adjacent sectors. A sector is called adjacent to another sector if both sectors contain a city network and there is more than one node that belongs to both sectors. For an example have a look at the traffic network shown in Figure 33. The only adjacent sector of sector 2 is sector 5. The only adjacent sector of sector 4 is sector 5. The adjacent sectors of sector 5 are the sectors 2 and 4. Sector 9 has no adjacent sector.

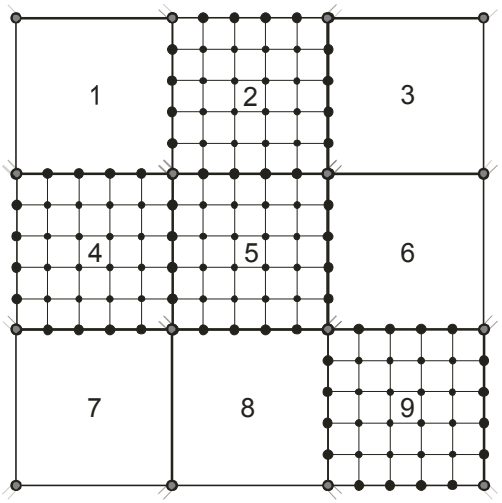

Figure 33: Network with adjacent sectors

Whenever there is an adjacent sector the routing tables of the Sector route finding system must contain a virtual node for this sector. This is necessary for the case where the destination sector of a requested route is an adjacent sector. In this case the vehicle should be routed directly to the adjacent sector and not to a motorway intersection first. But not all routing tables must contain a virtual node for every adjacent sector. The nodes of a sector

that also belong to an adjacent sector do not have a row for the virtual node of the adjacent sector in their routing table. This is because the adjacent sector is already reached.
To come back to the example in Figure 33 the routing tables of the intersections of sector 5 must contain a row for the virtual node of sector 2 and a row for the virtual node of sector 4. The routing tables of the slip roads/departures of sector 5 which also belong to sector 4 contain a row for the virtual node of sector 2 but not of sector 4. And for example the routing table of the motorway intersection of sector 5 which also belongs to the sectors 1, 2 and 4 contains no virtual node.
The update of the probabilities for the virtual nodes in the routing tables can be done by using forward agents whose destination is the virtual node. Such a forward agent randomly selects at each node the next node according to the magnitude of the probabilities in the row of the routing table which represents the virtual node. As soon as the forward agent arrives at a node of the sector that is represented by the virtual node (the forward agent's destination), the forward agent transforms to a backward agent because the destination is reached. Now the backward agent uses the memory of the forward agent to find the way back to the source node. At each node on its way back the backward agent updates the probabilities for the virtual node by using the travel times to the virtual node that are stored in the memory. So not only the probabilities in the routing table of the source node but also the probabilities in the routing table of each node on the path to the virtual node are updated. Besides at every node the backward agent updates not only the probabilities for the virtual node but also the probabilities for the destination node itself. And additionally the backward agent also updates the probabilities for every node on the sub paths to the destination node because the information for doing this is also available in the backward agent's memory. Figure 34 shows the described update procedure in pseudo code for backward agents of a Sector route finding system that update the probabilities for the virtual nodes. The procedure is executed by the backward agents at every node on the way back to the source node.

```
procedure UpdateRoutingTable(currentNode, memory)
  inputs: currentNode, the node where the backward agent is
          memory, the memory of the backward agent

  nextNode:=the successor node of currentNode taken from memory
  destinationNode:=the destination node of the route taken from memory
  virtualNode:= the adjacent destination sector of the forward agent taken from memory
  subpathNode:=currentNode
  travelTime:=0
  repeat
    subpathNode:=the successor node of suppathNode taken from memory
    travelTime:=travelTime + time to reach subpathNode from last node taken from memory
    update the probability P_subpathNode, nextNode in the routing table of currentNode with travelTime
  until subpathNode = destinationNode
  update the probability P_virtualNode, nextNode in the routing table of currentNode with travelTime
```

Figure 34: Update procedure for the backward agents which update the virtual nodes

5.4.1.3.2 Determining the agents' destination

What also should be adjusted for the Sector route finding systems is the determining of the forward agent's destination. The conventional procedure is to launch a forward agent at regular intervals from every node of the sector toward a randomly chosen destination node of the same sector. Now that there are also virtual nodes for the adjacent sectors in the routing tables not only nodes are a possible destination but also virtual nodes. So the possible destinations for the forward agents from each node are randomly chosen from all destinations that are contained in the routing table of the node.
Since in the Hierarchical routing system the vehicles are able to leave a sector, more forward agents should be launched to the motorway intersections than to the other nodes. The reason is that all the long-distance traffic that starts within a source sector and goes to a destination sector leaves the source sector by passing one of the four motorway intersections. This means that the routes to any possible destination in other sectors of the whole traffic network include the route from the source node to one of the four motorway intersections. Hence in a traffic network with a high percentage of long-distance traffic the attention should be directed more to the motorway intersections as destination of the forward agents of the Sector route finding systems. So the motorway intersections should have a higher probability to be chosen as destination of the forward agents than the other nodes in a sector. This results in more agents that update the probabilities of the motorway intersections in the routing tables. And therefore the optimization of the routes to the motorway intersections is improved.

5.4.1.3.3 Providing travel times

For the computation of the most suitable motorway intersection to go to destinations in other sectors a Sector route finding system needs the travel times to reach the four motorway intersections of the sector and the travel times to reach every destination of the abstract level from the four motorway intersections. The times to reach the motorway intersections are available in the time columns of all routing tables of the Sector route finding system. And the times to reach every possible destination of the abstract level from the motorway intersections are available in the time columns of the routing tables of the Global route finding system for the motorway intersections (see section 5.4.1.2). For the computation of the most suitable motorway intersection also the information is needed what node is the node with highest probability in each row of the routing table. With that information the next node to go to every destination of the abstract level is known. The reason why this information is needed is given in section 5.4.2.2.1.1, where the computation of the most suitable motorway intersection is considered.
It is not desirable to send a request to the Global route finding system every time some data from the routing tables of the motorway intersections is needed. Because the Hierarchical routing system is distributed there exist a communication delay for the communication between the distributed parts which can not be neglected. Every function call to another part of the system takes some time to receive the response. And such a delay to wait for data

from the Global route finding system would considerably slow down the performance of the Hybrid system.
To guarantee a fast access to the routing tables of the motorway intersections of the Global routing system, the routing tables should also be stored locally in the Sector route finding systems. So at every Sector route finding system a local copy of the routing tables of the four motorway intersections of the sector must exist. Because the bandwidth of the internet or a wide area network is limited, it is necessary to send as little information as necessary to allow a high update rate. For the Sector route finding systems it is not necessary to copy the whole routing table, what would mean a high amount of data which must be sent from the Global route finding system to the Sector route finding systems all the time. The information of the routing tables can be reduced by leaving out the probabilities. Only the information is necessary what node is the node with highest probability in every row. Therefore only a flag need to be set for every node.
Let's consider an example of such a reduced routing table. Table 8 that was introduced in section 5.4.1.2 (see page 48) is used as an example of a routing table of a motorway intersection of the Global route finding system. The corresponding, reduced routing table is shown in Table 12. Instead of a 32 Bit value for a floating point number to represent the probabilities, only one Bit is now needed to replace the probabilities. This is an enormous economy in needed space. If a 32 Bit value is assumed to represent the times in the time column the total economy for the entries of the whole routing table will be more than 75 percent. And if a 16 Bit value for the time values is assumed the economy will be even greater, namely more than 85 percent.

Table 12: Reduced routing table of a motorway intersection of the Global route finding system

Destination ↓ | Next node →	node 22	node 29	node 31	node 42	time
node 1	●				55 s
node 2	●				51 s
...					
node 22	●				6 s
…					
node 29		●			5 s
node 31			●		6 s
…					
node 42				●	7 s
…					
node 68				●	91 s
sector 3			●		29 s
sector 6			●		29 s

For the performance of the Hybrid system it is very important that every part of the Hierarchical routing system is provided with the newest information. Because the entries in routing tables change very quickly, possibly many times each second, the reduced routing tables of the motorway intersections, that are locally stored at the Sector route finding systems, need to be updated as quick as possible. So a high update rate is desirable. The possible update rate depends on the bandwidth of the connection and the amount of data that must be sent at every update step. For the update of the reduced routing tables a high update

rate can be ensured because the reduced routing tables must not be sent entirely at every update step. The flags for the rows need only be updated if another node becomes the node with highest probability in a row. This means as long as the probabilities change but the node with highest probability remains the same node the flags need not be updated.
For the time values an update measure can be introduced to avoid that all time values must be updated at every update step. The update measure has a numeric value which represents a change in per cent. At every update step a time value is only updated if it has changed more than the value of the update measure in per cent since it has been updated the last time. For example if the value of the update measure is 5, then a time value is only updated again if it has changed more than 5%. By that the Sector route finding systems must not be informed about every small change of a time value, what results in a great economy of data that must be sent at every update step.

5.4.2 Computation of the routes

Now that all necessary adjustments explained in section 5.4.1 have been introduced, the computation of the shortest routes for vehicles can be considered. An algorithm is introduced that computes the next steps of the route towards the destination for the next *n* seconds.
Cars that want to request a route connect with the responsible Route finding system according to the rules I and III that have been introduced in Figure 12 of section 3.2.4. So if a car is inside a sector it will connect with the Sector route finding system of that sector (rule I). And if a car is on a motorway it will connect with the Global route finding system (rule III). But applying these rules does not always mean for the vehicle to be connected with the responsible Route finding system. For instance consider the case where a car is still in the source sector but on a motorway when it wants to request the next part of the route to the destination. The motorway of the source sector can be used by the Sector route finding system to route the vehicle to the most suitable motorway intersection. By following rule III the car must connect with the Global route finding system because the car is on a motorway. But in fact the Sector route finding system of the source sector is still responsible because it must route the vehicle to the most suitable motorway intersection first. So the Route finding systems must always consider whether they are responsible or not. If not they will have to determine and ask the responsible Route finding system to compute the route. After the route is computed the responsible Route finding system returns the route to the Route finding system the car is connected with. And this system sends the route to the vehicle.

5.4.2.1 Communication interfaces

The Sector route finding systems and the Global route finding system each provide a function which vehicle must call to request a route. These functions constitute the interface for the communication between the cars and the Route finding systems. Figure 35 shows the function of the Sector route finding systems that vehicles will have to call if they are inside a sector (rule I). And Figure 36 shows the function of the Global route finding system that vehicles will have to call if they are on a motorway (rule III). In both functions the vehicles

need to provide four parameters. These are the current position of the vehicle (*currentNode*), the destination of the vehicle (*destinationNode*), the sector where the vehicle started (*sourceSector*) and the number of seconds for which the route shall be computed beforehand (*time*). The two functions call the function computeRoute of the same Route finding system to compute the route. The return parameter *route* is a list that contains the next links of the route towards the destination for the given time.

function SectorRFS.requestRoute(*currentNode*, *destinationNode*, *sourceSector, time*) **returns** *route*
inputs: *currentNode*, the node where the car actually is
destinationNode, the destination node of the route to plan
sourceSector, the sector where the vehicle started
time, the length of the route in seconds to be computed

route:=an empty List
memory:= a datastructure with the two fields *memory.nodeToLeaveSector* and *memory.adjacentSector*, both fields are still empty
timeLeft:=*time*
return *thisSectorRFS*.computeRoute(*currentNode*, *destinationNode*, *sourceSector*, *route*, *memory*, *timeLeft*)

Figure 35: Function requestRoute of the Sector route finding systems

function GlobalRFS.requestRoute(*currentNode*, *destinationNode*, *sourceSector, time*) **returns** *route*
inputs: *currentNode*, the node where the car actually is
destinationNode, the destination node of the route to plan
sourceSector, the sector where the vehicle started
time, the length of the route in seconds to be computed

route:=an empty List
memory:= a datastructure with the two fields *memory.nodeToLeaveSector* and *memory.adjacentSector*, both fields are still empty
timeLeft:=*time*
return *globalRFS*.computeRoute(*currentNode*, *destinationNode*, *sourceSector*, *route*, *memory*, *timeLeft*)

Figure 36:Function requestRoute of the Global route finding system

By setting the parameter *time* to a low value, for example 1 second, a Route finding system will always compute only the next link to go to. This means that the vehicles will have to request the next link at every intersection. For performance issues the time parameter should be set to a higher value so that the vehicles have to ask for the next links at the earliest after using several links. By setting the parameter *time* to a very high value the Route finding system will compute and return the complete route to the destination node. That means that the vehicle will have to request the route only once. But this results in a fixed route that is not updated during the travel to the destination. And so no adaptation of the route according to a new traffic situation is possible. So the parameter time should have a well chosen value to avoid too much communication and computation on the one hand and to facilitate a fast adaptation to new traffic conditions on the other hand.
The parameter *sourceSector* is very important for the computation of the routes that start within one sector and that end within another sector. This value represents the sector where the route starts and must be stored by the vehicle. In cases where the vehicles do not leave

the sector or where they start to request a route when they are already on a motorway, a null value can be used for this parameter.

5.4.2.2 Functions computeRoute

Let's consider the functions computeRoute of the Sector route finding systems and the Global route finding system. These functions are recursive functions which call themselves recursively. In each run of the function the next link of the route to the destination node is computed and added to the list *route* if the Routing system is responsible for the computation. Else the responsible Routing system is determined and engaged to compute the next steps of the route.

5.4.2.2.1 Function SectorRFS.computeRoute

Figure 37 shows the algorithm of the function computeRoute of the Sector route finding systems in an object oriented style. A flow chart of this function, which visualizes the structure of the algorithm, is given in Figure 38. The numbers in the flow chart represent the corresponding program lines from the algorithm in Figure 37. Note that the flow chart is referred to one run of the recursive function SectorRFS.computeRoute where the next link of the route is determined or another Routing system is engaged to compute the route.
The function SectorRFS.computeRoute distinguishes mainly two different cases. Line 13 contains the if-statement that determines which case applies. The first case is the instance where a vehicle is within a city network and the destination node is in the same sector. In this case the vehicle must be routed to the destination node by the Sector route finding system of the sector where the destination node is in. This instance is the if-branch in lines 14 till 26. The second case is the instance where a vehicle is within a city network and the destination node is a node of another sector. In that case the vehicles must be routed to the most suitable motorway intersection of the source sector or to an adjacent sector by the Sector route finding system. That case is the else-branch in lines 27 till 54.

Let's consider the first case now. The test in line 15 whether the destination node belongs to the sector of the Sector route finding system is necessary for one special case. This is the case where a vehicle is routed to the adjacent sector because the destination node is a node in the adjacent sector. At the moment when the car enters the adjacent sector the if-statement applies and the Sector route finding system of the adjacent sector assumes to compute the rest of the route. In section 5.4.2.4 an example will be given to illustrate this case.
The function nextNode in line 19 plays a very important role. This is the function that computes the next node for the route to the destination node. It uses the routing table of the node that is handed over by the first parameter. In that routing table it looks in the row that is determined by the second parameter, the destination node. The function returns the node which has the highest probability in the row. And this node becomes the next node of the route. So at this place the data in the routing tables that is computed and updated by the agents is used to determine the next node. By deciding for the node with highest probability

the cars are always routed along the path with highest probability. And this is the core idea of the ABC-algorithm.

```
01  function SectorRFS.computeRoute(currentNode, destinationNode, sourceSector, route, memory, timeLeft) returns route
02     inputs: currentNode, the node to which the route have been planned till now
03             destinationNode, the destination node of the route to plan
04             sourceSector, the sector where the vehicle started
05             route, a list containing the so far computed route
06             memory, a data structure to deliver necessary information
07             timeLeft, the time left for the rest of the route
08
09     currentNodeSectors:=currentNode.getSectors()
10     destinationNodeSectors:=destinationNode.getSectors()
11     commonSectors:=currentNodeSectors ∩ destinationNodeSectors
12     //belong currentNode and destinationNode to the same sector?
13     if(commonSectors.size()>0) then
14        //is the sectorRFS responsible? test necessary for case with adjacent sectors
15        if(not isNodeOfThisSector(destinationNode)) then
16           responsibleRFS=getRFS(destinationNode.getSector())
17           return responsibleRFS.computeRoute(currentNode, destinationNode, sourceSector, route, memory, timeLeft)
18        //use the routing table to lookup the next node to go to
19        nextNode:=nextNode(currentNode, destinationNode)
20        link:=getLink(currentNode, nextNode)
21        route.add(link)
22        timeLeft:=timeLeft-link.getLength()/link.getAllowedSpeed()
23        if(nextNode=destinationNode or timeLeft<=0)
24           return route
25        else
26           return thisSectorRFS.computeRoute(nextNode, destinationNode, sourceSector, route, memory, timeLeft)
27     else
28        //the vehicle is still in the source sector
29        if(memory.nodeToLeaveSector=null and memory.adjacentSector=null) then
30           //are currentNode and destinationNode in adjacent sectors?
31           if(inAdjacentSectors(currentNode, destinationNode) and destinationNode.isIntersection()) then
32              memory.adjacentSector:=destinationNode.getSector()
33           else
34              if(destinationNode.isIntersection()) then
35                 destination=destinationNode.getSector()
36              else
37                 destination=destinationNode
38              //compute the most suitable motorway intersection
39              memory.nodeToLeaveSector:=computeMotorwayIntersectionForShortestRoute(currentNode, destination)
40        //is the motorway intersection reached?
41        if(currentNode=memory.nodeToLeaveSector) then
42           return globalRFS.computeRoute(currentNode, destinationNode, sourceSector, route, memory, timeLeft)
43        //use the routing table to lookup the next node to go to
44        if(memory.nodeToLeaveSector<>null) then
45           nextNode:=nextNode(currentNode, memory.nodeToLeaveSector)
46        else
47           nextNode:=nextNode(currentNode, memory.adjacentSector)
48        link:=getLink(currentNode, nextNode)
49        route.add(link)
50        timeLeft:=timeLeft-link.getLength()/link.getAllowedSpeed()
51        if(nextNode=destinationNode or timeLeft<=0)
52           return route
53        else
54           return thisSectorRFS.computeRoute(nextNode, destinationNode, sourceSector, route, memory, timeLeft)
```

Figure 37: Function computeRoute of the Sector route finding systems

In line 20 the function getLink is used to get the link that connects the current node and the next node. This link is added to the end of the *route*. Now the remaining time *timeLeft* for the rest of the route is computed in line 22 by subtracting the optimal time needed to cross the link from the current value of the variable *timeLeft*. If the destination node is reached or no time is left the computed route is returned and the execution of the function is finished (lines 23 and 24). Else the route is lengthened by at least one further node by recursively calling the function computeRoute of the same Sector route finding system again (line 26).

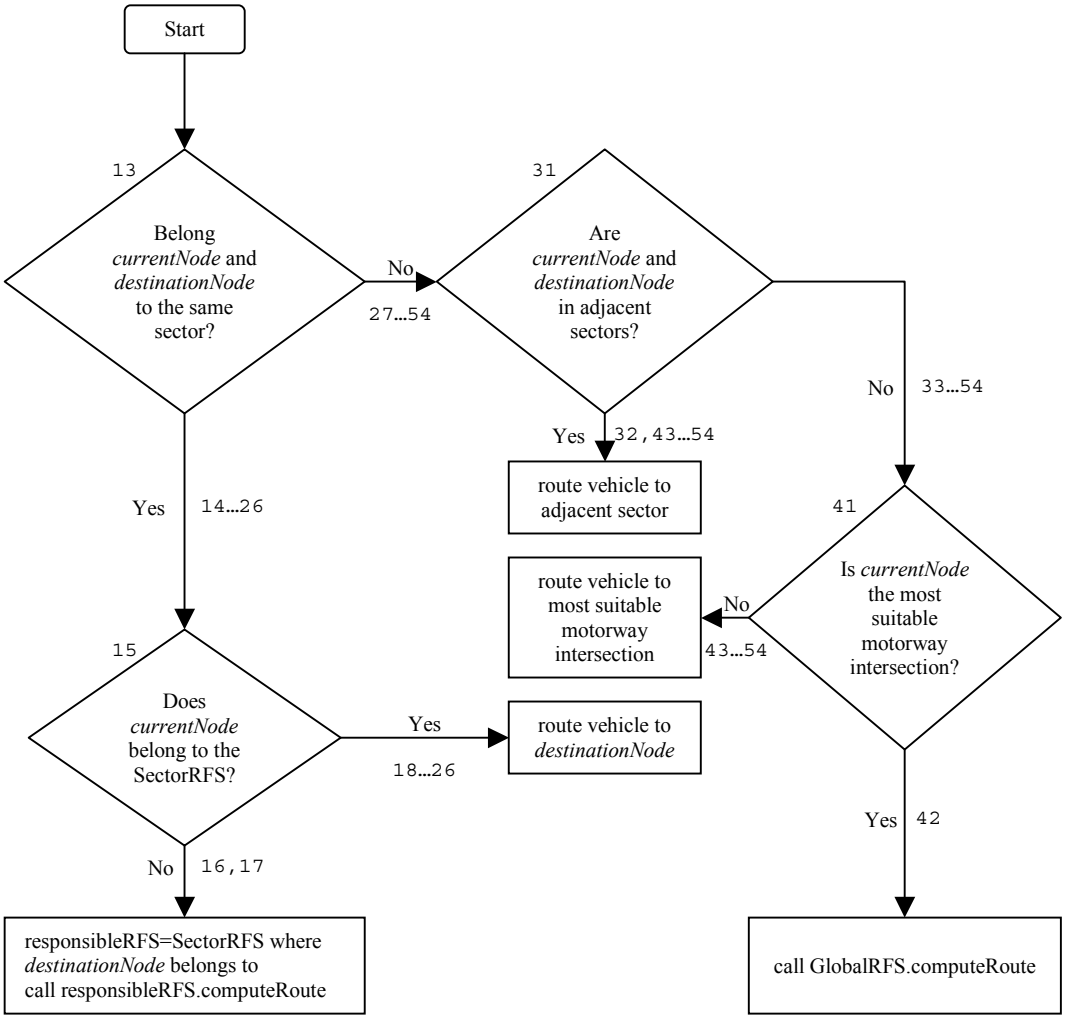

Figure 38:Flow chart of the Function computeRoute from Figure 37

Let's consider the case where the vehicle is within a city network and the destination is in another sector (lines 27 till 54). This case is more complicated than the first one. The data structure *memory* is used to store the next destination where the Sector route finding system must route the car to. In line 29 it is checked whether one of the fields of the data structure *memory* has been set so far. This will not be the case if the function computeRoute is executed the first time in the tree of recursive function calls. So in that case one of the two

fields is initialized with a value. If the destination node is in an adjacent sector the field *adjacentSector* is set to the sector of the destination node (line 32) and the car must be routed to the adjacent Sector. Else the field *nodeToLeaveSector* is set to the most suitable motorway intersection along which the sector must be left by using the function computeMotorwayIntersectionForShortestRoute (line 39). This function takes two parameters, the current position of the route and the destination. The destination can be a sector or a node of the abstract level. If the destination node is an intersection what means that it is a node within a sector the parameter destination will be set to the sector where the destination node belongs to (lines 34, 35). And if the destination node is a node of the abstract level the parameter destination will be set to the destination node (line 37). With the two parameters the function computeMotorwayIntersectionForShortestRoute computes the most suitable motorway intersection which then is returned. This function is very important and is explained in detail in the next section, section 5.4.2.2.1.1.
In line 41 it is checked whether the route has reached the most suitable motorway intersection. If this occurs the source sector will be left by continuing the travel along the motorway. So the Global route finding system is responsible to compute the next part of the route to the destination. That is realized by calling the function computeRoute of the Global route finding system in line 42.
In the following lines the Sector route finding system computes the next node of the route toward the next destination which is stored in the data structure memory by using the routing table of the current node. If the destination node is not in an adjacent sector the route will have to be planned to the most suitable motorway intersection. So the next node is determined by using the row in the routing table that represents the most suitable motorway intersection (line 45). But if the destination node is in an adjacent sector the route will have to be planned to the adjacent sector. This is done in line 47 by using the row in the routing table that represents the virtual node of the adjacent sector.
In the lines 48 till 54 the same happens as in the lines 20 till 26. The link to the next node is added to the route and the value of *timeLeft* is updated. The route will be returned if the destination node is reached or no time is left. Else the next node of the route is computed by calling the function computeRoute of the same Sector route finding system again (line 54).

5.4.2.2.1.1 Computation of the most suitable motorway intersection

The most suitable motorway intersection is computed by the function computeMotorwayIntersectionForShortestRoute (see function SectorRFS.computeRoute in line 39). It considers the four routes from the current node (first parameter) to the destination (second parameter) along one of the motorway intersections of the source sector. The destination can be another sector represented by its virtual node, a slip road/departure or a motorway intersection of another sector.
In the first step of the computation the four motorway intersections of the source sector are investigated whether they are a possible candidate for the most suitable motorway intersection. The most suitable motorway intersection must be a node along which the source sector is left to go to the destination. For the motorway intersections of the source sector this depends on the further route to the destination which is computed by the Global route finding system. So a motorway intersection is a candidate for becoming the most suitable motorway intersection if the next node toward the destination is a node of another

sector. But if the next node is a node of the source sector then the motorway intersection is not a candidate for the most suitable motorway intersection. In this case the Global route finding system would route the vehicle from this motorway intersection along another motorway intersection of the source sector to go to the destination. And this is not desirable because the Sector route finding system should route the car to the motorway intersection along which the source sector will be left. In this way the streets of the source sector can also be used to find the shortest route to the motorway intersection.

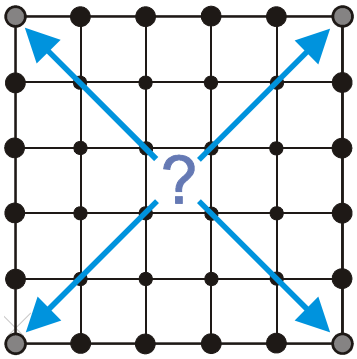

Figure 39: What is the most suitable motorway intersection?

The motorway intersections which are not a candidate for the most suitable motorway intersection must be filtered out in the first step. Therefore the locally stored, reduced routing tables of the abstract level of the four motorway intersections of the sector are needed (see section 5.4.1.3.3). In the reduced routing tables the information is contained what neighboured node of the motorway intersection is the next node to the destination. This is the node whose flag is set in the row that represents the destination. If the next node is a node of the source sector the motorway intersection will be filtered out.
For the remaining motorway intersections the travel times from the current node along the motorway intersection to the destination are computed. This is done by using the time values in the time column of the routing tables. So for example the travel time of the route from the current node along one of the remaining motorway intersections to the destination is the time to reach the motorway intersection from current node plus the time to reach the destination from the motorway intersection. The time to reach the motorway intersection is contained in the time column of the routing table of the Sector route finding system for the current node (see section 5.4.1.2). And the time to reach the destination from the motorway intersection is contained in the locally stored, reduced routing tables of the abstract level of the motorway.
After the travel times for the routes are computed the route with the shortest travel time is selected. The function computeMotorwayIntersectionForShortestRoute finally returns the motorway intersection along which the selected route goes. This is the most suitable motorway intersection that must be used to leave the sector.

5.4.2.2.2 Function GlobalRFS.computeRoute

Now the function computeRoute of the Global route finding system is considered. Figure 40 shows the source code of this function in an object oriented style and Figure 41 displays the flow chart of this function. The numbers in the flow chart represent the corresponding program lines from the algorithm in Figure 40. The flow chart is referred to one run of the recursive function computeRoute.

```
01  function GlobalRFS.computeRoute(currentNode, destinationNode, sourceSector, route, memory, timeLeft)
02  returns route
03     inputs: currentNode, the node to which the route have been planned till now
04             destinationNode, the destination node of the route to plan
05             sourceSector, the sector where the vehicle started
06             route, a list containing the so far computed route
07             memory, a data structure to deliver necessary information
08             timeLeft, the time left for the rest of the route
09
10     currentNodeSectors:=currentNode.getSectors()
11     destinationNodeSectors:=destinationNode.getSectors()
12     commonSectors:=currentNodeSectors ∩ destinationNodeSectors
13     //is the destination sector reached?
14     if(commonSectors.size()>0 and destinationNode.isIntersection()) then
15        responsibleRFS:=getRFS(destinationNode.getSector())
16        return responsibleRFS.computeRoute(currentNode, destinationNode, sourceSector, route, memory, timeLeft)
17     else
18        //the destination sector has not been reached yet
19        //is the vehicle still in the source sector?
20        if(currentNodeSectors.contains(sourceSector) and currentNode<>memory.nodeToLeaveSector)  then
21           responsibleRFS:=getRFS(sourceSector)
22           return responsibleRFS.computeRoute(currentNode, destinationNode, sourceSector, route, memory, timeLeft)
23        if(destinationNode.isIntersection()) then
24           destination:=destinationNode.getSector()
25        else
26           destination:=destinationNode
27         //use the routing table to lookup the next node to go to
28        nextNode:=nextNode(currentNode, destination);
29        link:=getLink(currentNode, nextNode)
30        route.add(link)
31        timeLeft:=timeLeft-link.getLength()/link.getAllowedSpeed()
32        if(nextNode=destinationNode or timeLeft<=0)
33           return route
34        else
35           return globalRFS.computeRoute(nextNode, destinationNode, sourceSector, route, memory, timeLeft)
```

Figure 40: Function computeRoute of the Global route finding system

This function distinguishes three different cases. The first case is the instance where the destination node is a node within a sector (an intersection) and the sector, to which the destination node belongs to, is reached (if-statement in line 14). In that case the last part of the route to the destination node must be computed by the Sector route finding system where the destination node belongs to. In line 15 this Sector route finding system is determined and in line 16 the function computeRoute of this system is called.

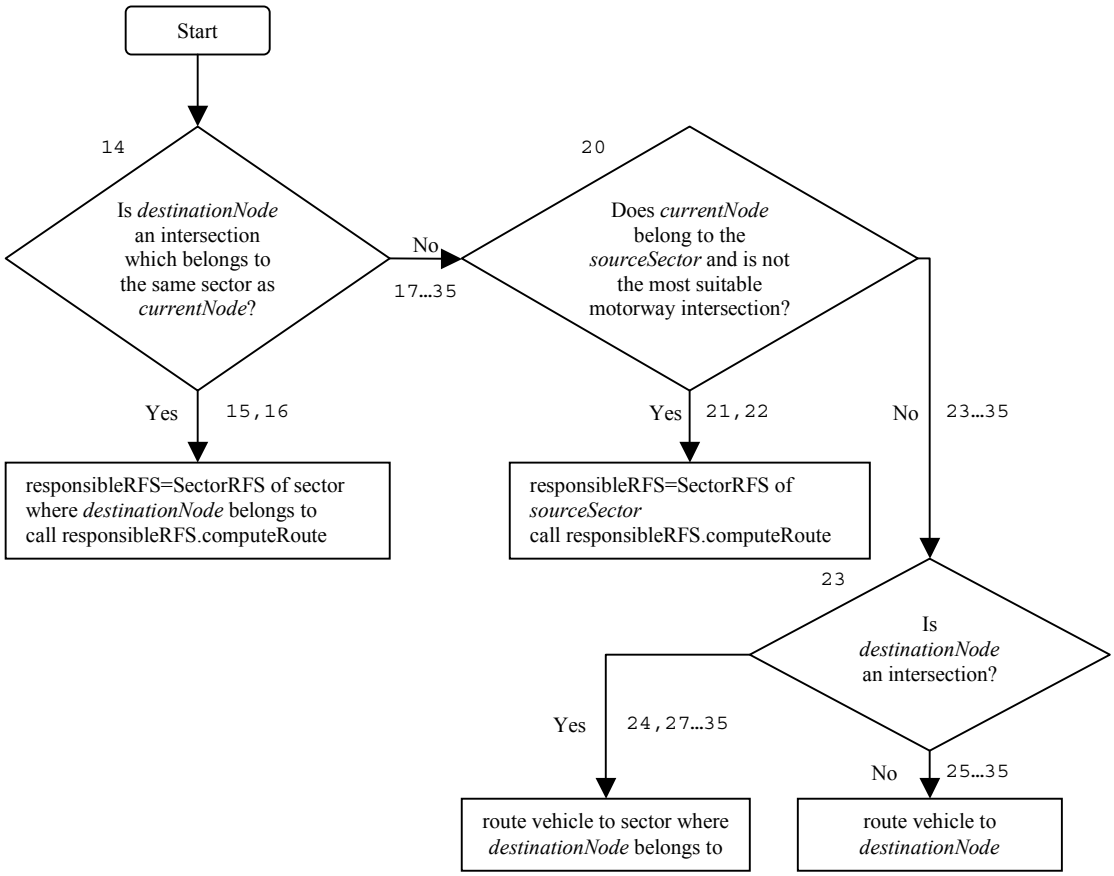

Figure 41: Flow chart of the Function computeRoute from Figure 40

The else-branch in lines 17 till 35 represents the instance that the destination sector has not been reached yet and/or that the destination node is not an intersection but a node of the motorway. Here a second case must be considered. Line 20 contains the if-statement that checks whether the current node still belongs to the source sector and whether the current node is not the most suitable motorway intersection along which the source sector must be left. If this is the case then the Sector route finding system of the source sector is still responsible to compute the route to the most suitable motorway intersection of the source sector. Therefore in line 21 the Sector route finding system is determined and in line 22 this system is engaged to compute the next step(s) of the route.
The lines 23 till 35 represent the third case where neither the Sector route finding system of the source sector nor the Sector route finding system of the destination sector but the Global route finding system is responsible to compute the next part of the route. In lines 23 till 26 the destination of the next part of the route is determined. If the destination is a node within a sector the destination will be the sector to which the destination node belongs to (line 24). And if the destination is a node of the abstract level then the destination will be the destination node (line 26). The function nextNode is applied in line 28 to determine the next node toward the destination by using the routing table of the current node. The link to the next node is added to the route and the value of *timeLeft* is updated. If the destination is

reached or no time is left the route is returned (line 33). Else the function computeRoute of the Global route finding system is executed another time.

5.4.2.3 Example 1: Distant sector

In this section an example is given how the cooperation of the Route finding systems for the route computation functions for computing a route from a node within a sector to a node within a distant sector. Figure 42 shows a traffic network with nine sectors where five of them contain a city network. In this network a route of a vehicle from the source node (node 1) to the destination node (node 20), which was routed by the Hierarchical routing system, is displayed. The dark grey and the light grey sections of the route each represent one part of the route that was computed by calling the function requestRoute of the corresponding Route finding system. So with each execution of the function requestRoute the next three or four links of the route to the destination node were computed. This is because of the choice of the parameter *time*. The route request points are the locations where the car requested the next part of the route. So in the example the car did a route request six times on the way to the destination node.

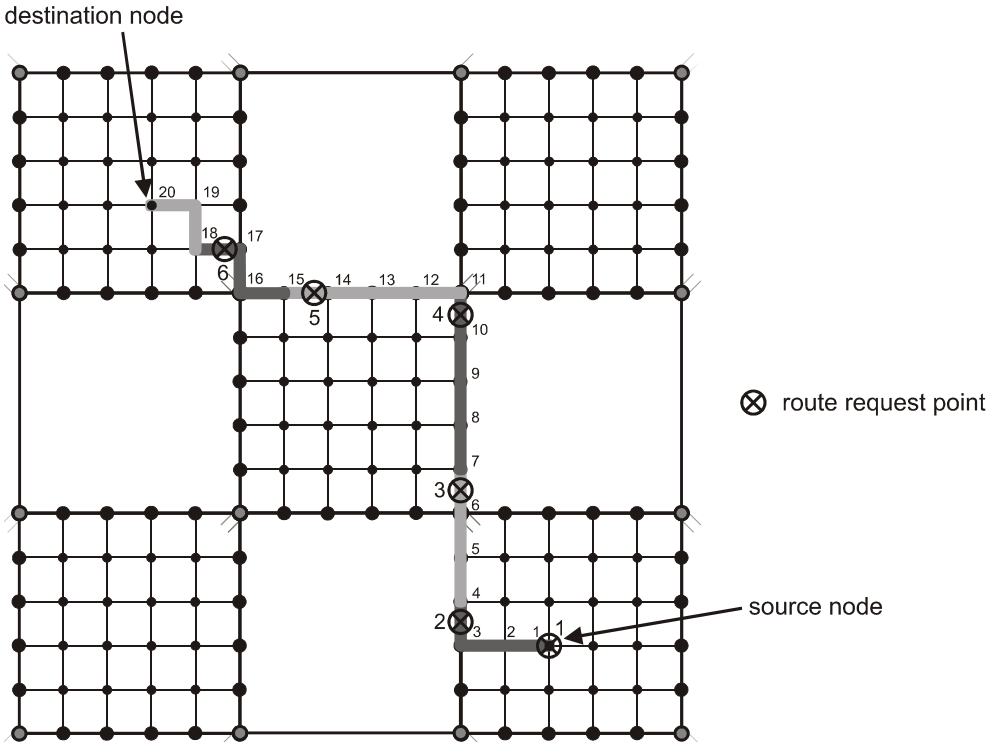

Figure 42: A car's route to an intersection in a distant sector

In the following the execution of the function requestRoute at every route request point is described.

Route request point 1:

Let's start at the moment where the car is at the source node. The car stores the number of the sector, where it actually is, locally in its memory. Every time when the car calls the function requestRoute it hands over this locally stored sector number via the parameter *sourceSector*. Because the car is on a street inside a sector it connects with the Sector route finding system of the source sector to request the route. Connected with this Sector route finding system the car calls the function SectorRFS.requestRoute which then calls the function SectorRFS.computeRoute of the same Sector route finding system. The else-branch in lines 27 till 54 of the function SectorRFS.computeRoute is entered because the vehicle is in the source sector and not yet in the destination sector. Because the destination node is not in an adjacent sector the most suitable motorway intersection is computed and stored in the data structure memory (lines 34 till 39). This is node 6. Now the next node on the path to the most suitable motorway intersection is determined with the function nextNode by using the routing table of node 1. That is node 2. The link to this node is added to the empty list route. The parameter *timeLeft* is updated by subtracting the optimal time that is needed to go to node 2. Because the value of *timeLeft* is still greater than zero the function computeRoute of the same Sector route finding system is executed a second time.
In this second run the most suitable motorway intersection is not computed again because it was already computed in the last run and it is stored in the data structure memory. So this node is used as most suitable motorway intersection and node 3 is determined as the next node toward the most suitable motorway intersection. The link from node 2 to node 3 is added to the list *route*.
The same repeats in the third run, namely the link from node 3 to node 4 is added to the list *route*. But now the parameter *timeLeft* is smaller than zero. So in the third run of the function SectorRFS.computeRoute the route is returned and no further recursive function calls are done. The first part of the route that consists of the three links between the nodes 1, 2, 3 and 4 is sent to the vehicle. The vehicle starts and follows the route.

Route request point 2:

Before the vehicle arrives at node 4 it makes a second route request because it needs the further steps of the route. But now the vehicle is on a motorway and so it connects with the Global route finding system. But because the vehicle is still in the source sector the Global route finding system forwards the route request to the Sector route finding system of the source sector (see function GlobalRFS.computeRoute in line 20 till 22). The function SectorRFS.computeRoute must compute the most suitable motorway intersection to go to the destination sector. Because the traffic situation has not changed dramatically since the last route request node 6 is again determined as the most suitable motorway intersection. Now the link from node 4 to node 5 is added to the route and the function computeRoute is called again.
In this second run the link to node 6 is added to the route. Again the function SectorRFS.computeRoute is called recursively. But now in the third run the current node is the most suitable motorway intersection along which the source sector must be left. The if-statement in line 41 in the function SectorRFS.computeRoute evaluates to true and the Sector route finding system engages the Global route finding system to compute the next

steps of the route. Figure 43 shows the resulting tree of the recursive function calls from the route request at point 2.
The function GlobalRFS.computeRoute now computes the next link toward the destination sector by using the routing table of the abstract level of node 6. This is the link from node 6 to node 7. The link is added to the route and the parameter *timeLeft* is updated. Because the value of *timeLeft* is smaller than zero the computed route is returned to the function SectorRFS.computeRoute in level 5 of the tree of the recursive function calls. From there it is returned to level 4, then to level 3, 2 and to level 1. Finally the function GlobalRFS.requestRoute sends the route, which was partly computed by the Sector route finding system and partly by the Global route finding system, back to the vehicle. The vehicle continues its travel to the destination node by following the computed links.

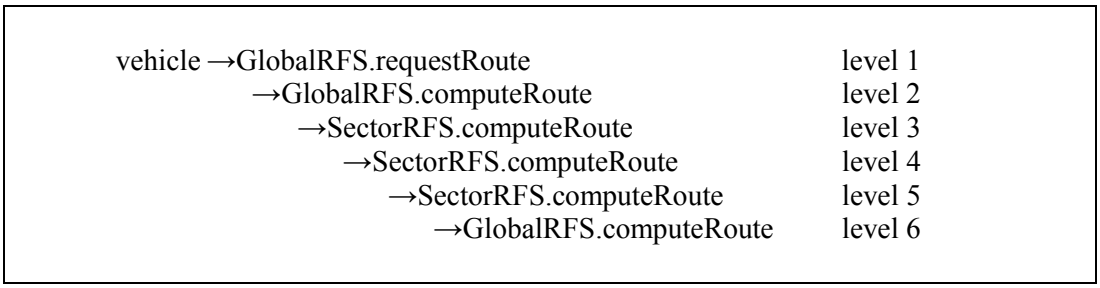

Figure 43: Tree of the recursive function calls

Route request point 3:

At route request point 3 the car makes a new route request. It connects with the Global route finding system which is responsible for the route computation. The function GlobalRFS.computeRoute is executed four times. In each run the next node toward the destination sector is computed and added to the list *route*. The route which is sent to the vehicle after the fourth run of the function goes along the nodes 8, 9, 10 and 11.

Route request point 4:

At route request point 4 the same happens like at the last route request point. The vehicle connects to the Global route finding system. The Global route finding system computes the next four links of the route by executing the function GlobalRFS.computeRoute and returns it to the vehicle.

Route request point 5:

When the vehicle arrives at route request point 5 it again connects with the Global route finding system. The Global route finding system adds the link from node 15 to node 16 to the empty list *route* in the first run of the function GlobalRFS.computeRoute.
In the second run of the function GlobalRFS.computeRoute the if-statement in line 14 evaluates to true because the destination sector is reached. Now the Sector route finding system of the destination sector is engaged to continue the route computation by calling the function SectorRFS.computeRoute of this system (lines 15, 16).

In the execution of the function SectorRFS.computeRoute the if-statement in line 13 evaluates to true because the destination node is in the same sector as the current node (node 16). The Sector route finding system uses its routing table of node 16 to determine the next node toward the destination node. This is node 17.
The function SectorRFS.computeRoute is executed a second time during which the link from node 17 to node 18 is added to the list *route*. Now the value of the parameter *timeLeft* is smaller than zero. So the computed route is returned and sent to the vehicle.

Route request point 6:

At route request point 6 the vehicle does the last route request on the way to the destination node. It connects with the Sector route finding system of the destination sector. The function SectorRFS.computeRoute is executed two times. In the fist run the link from node 18 to node 19 is added to the empty list *route*.
In the second run the link to node 20 is added. Now the if-statement in line 23 evaluates to true because the destination node is reached. So the route is returned and sent to the car. Now the car follows this last part of the route to reach the destination node.

5.4.2.4 Example 2: Adjacent sector

Now an example is given for routing a car to an adjacent sector. Figure 44 shows two adjacent sectors and the route of a car from a source node in the left sector to a destination node in the adjacent sector. This route was computed by the Hierarchical routing system.

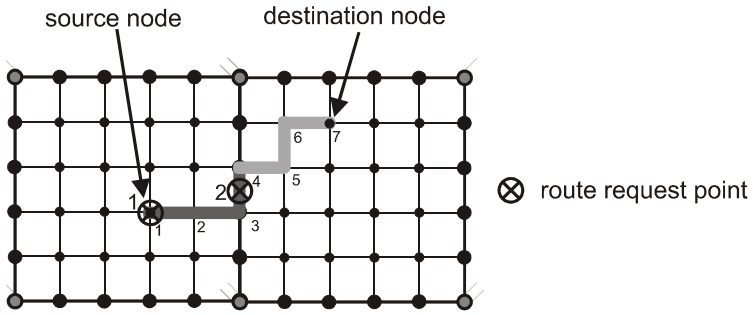

Figure 44: A car's route to an intersection in an adjacent sector

Let's consider the route computation process for this example:

Route request point 1:

The car starts the route at node 1. It connects with the Sector route finding system of the source sector and calls the function SectorRFS.requestRoute. In that way the function SectorRFS.computeRoute is executed. The else-branch in lines 27 till 54 of the function SectorRFS.computeRoute is entered because the vehicle has not yet arrived at the destination sector. Because the destination node is in an adjacent sector the field *adjacentSector* of the data structure *memory* is set to the sector where the destination node

belongs to (line 32). Using the routing table of node 1 the next node toward the adjacent sector is determined. This is node 2. The link from node 1 to node 2 is added to the empty list *route*. The function SectorRFS.computeRoute is called a second time.
In this second run the next node toward the adjacent sector which is taken from the data structure memory is determined. This is node 3. So the link that connects node 2 with node 3 is added to the list route. Now the function SectorRFS.computeRoute of the same Sector route finding system is executed again.
In this third run the if-statement in line 13 is evaluated to true because the destination sector is reached. But the Sector route finding system of the source sector is no more responsible for the further route computation. So the function SectorRFS.computeRoute of the Sector route finding system of the destination sector is called in line 17.
Now this Sector route finding system computes the next node of the route to the destination node. The next node of the route is node 4. The corresponding link is added to the list *route*. Now the value of the parameter *timeLeft* is smaller than zero. So the route is returned and sent to the vehicle. The vehicle now follows this computed part of the route.

Route request point 2:

When the vehicle is on the link to node 4 it again makes a route request to the Hierarchical routing system to request the next steps of the route toward the destination node. Because the car is on a motorway it connects with the Global route finding system. So the function GlobalRFS.computeRoute is executed. But because the destination sector is already reached the function GlobalRFS.computeRoute calls the function SectorRFS.computeRoute of the Sector route finding system of the destination sector in line 16.
The function SectorRFS.computeRoute then is recursively called three times and computes the route along nodes 5 and 6 to node 7. Because the destination node is reached in the third run the route is returned and sent to the car. The car now knows the last part of the route to go to the destination node.

Chapter 6 : Evaluation

In this last chapter the developed Hierarchical routing system is validated and recommendations are given how it can be extended and enhanced in the future. Because no running implementation exists yet it is not possible to draw conclusions from the experiments with a prototype. So more or less, only expectations can be formulated in this chapter.

6.1 Validating the model

Here in this section the model of the Hierarchical routing system is validated by considering the criteria optimality of the routes, scalability, real time aspect, robustness and applicability for realistic traffic networks. But first a short summary of the mode of operation of the Hierarchical routing system is given.
For the route optimization and route computation the Hierarchical routing system uses computation only and no reasoning. Under the condition that each of the Routing systems, which are distributed on separate computer systems, runs on one computer system and is not distributed itself, the design of the Hierarchical routing system ensures as less communication as possible. There is still a big amount of data which must be sent especially from the Global routing system to the Sector routing systems. But all agents, the forward, backward and time agents, belong to exactly one Routing system and so they need not be transferred to other computer systems. And all necessary information for the route optimization is locally available at each Routing system. By that the route optimization is not slow down by communication with other computer systems and waiting for answers. Only for the route computation message passing to other Routing systems and waiting for answers can be necessary. But it will be explained that this message passing should not disturb the real time aspect.

6.1.1 Optimality of the routes

As already explained in section 1.2.2 the ABC-algorithm is not optimal what means that there is no guarantee for finding absolute shortest paths in time. By that the routes for individual car drivers, which are computed by the Route finding systems, are not guaranteed to be optimal. But the routes computed by the ABC-algorithm are at least supposed to be close to the optimum and do not deviate strongly. The strong point of the ABC-algorithm is that it routes the whole traffic together in an optimal way by better exploitation the capacity of the road network. In that way all road user can profit from less congestions and shorter travel times.
The main question for this section is what losses in optimality of the routes must be accepted due to the hierarchy compared with routing in a non hierarchical network using the conventional ABC-algorithm. By the introduction of the hierarchy the whole traffic network is split up into several smaller networks, the motorway network (abstract level) and the city networks (detailed level). The Global routing system is responsible for the abstract level and

the Sector routing systems are each responsible for one city network. That means the responsibility and the focus of the Routing systems is restricted to one of the smaller networks. In that way the Routing systems can only include the nodes and links of their network to route vehicles in shortest time to a destination. And this results in some risks not to be able to find the shortest route in time. In the following sections the cases are explained where the Hierarchical routing system is not capable to find the optimal route.

6.1.1.1 Routing within a city network

In the case, where a vehicle is inside a sector and wants to be routed to another node inside the same sector, the Sector route finding system of the sector computes the shortest route. All other nodes and links of the sector including the surrounding motorway ring are concerned by the Sector route finding system to reach the destination node. In most of the cases the optimal route should run inside the sector only and so the Sector route finding system is capable to find it. But it is also possible that the shortest route in time contains nodes and links of an adjacent city network. Figure 45 shows an example for that case.

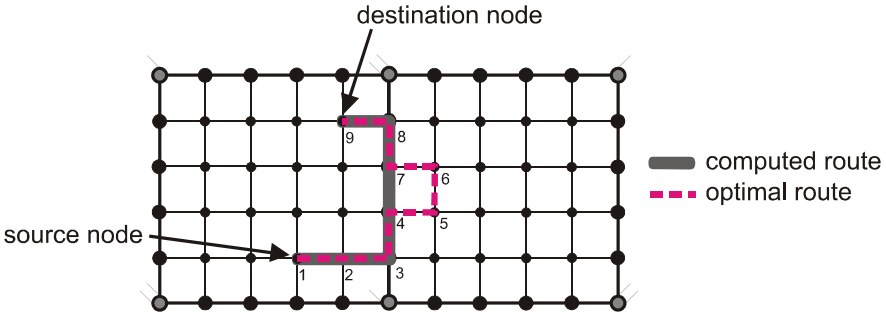

Figure 45: Route within a city network

The route, which is computed by the Sector route finding system of the left sector, starts at the source node, goes along the nodes 2 and 3 to the motorway, continues on the motorway along nodes 4, 7 and 8 and goes then to the destination node. But indeed the optimal route in time contains a small detour through the adjacent sector, because the link between nodes 4 and 7 is congested. The Hierarchical routing system is not capable to find this route.

6.1.1.2 Adjacent sector

If a vehicle is within a sector and the destination is a node in an adjacent sector the vehicle is routed on the shortest route in time to the adjacent sector and from there on the shortest route in time to the destination node. But the resulting composed route is not necessarily the shortest route in time, because when the car is routed to the adjacent sector, the direction, where the destination node is, is not taken into consideration. Figure 46 shows an example for a non optimal route to a node in an adjacent sector.

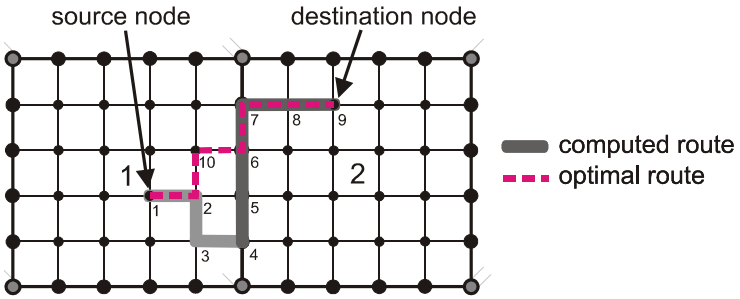

Figure 46: Route to an adjacent sector

The Sector route finding system of sector 1 computed the light grey part of the route and the Sector route finding system of sector 2 the dark grey part. The shortest route to the adjacent sector in this example is the path along nodes 2 and 3 to node 4. But for reaching the destination node this path is a detour. The optimal route to the destination node goes along node 10, what is the direction of the destination node. But because the Sector route finding system of sector 1 has no information about the direction of the destination node in sector 2 it can only route the vehicle to the adjacent sector. And this is in the example far away from the shortest route in time.

6.1.1.3 Distant sector

Vehicle, which need to be routed from a street inside a sector to a street inside a distant sector, are routed along the motorway from the source sector to the destination sector. Between both sectors only the motorway is considered and no streets of a city network are used. By that the possibilities for routes to the destination sector are restricted to the motorways. But if the motorways are congested, then it might be shorter to go through another city network to avoid the congestion.
An example is shown in Figure 47. The route computed by the Hierarchical routing system consists of three parts. Part 1, the shortest route in time from the source node to the most suitable motorway intersection of the source sector is computed by the Sector route finding system of the source sector. Part 2, the shortest route in time along the motorway to the closest motorway intersection of the destination sector, is computed by the Global route finding system. And part 3, the shortest route in time from that motorway intersection to the destination node is computed by the Sector route finding system of the destination sector. Nevertheless the composed route is not the optimal route because part 2 is restricted to the motorways only. The optimal route goes through the middle city network to avoid a congestion on the motorway. But the Hierarchical routing system is not able to find it.

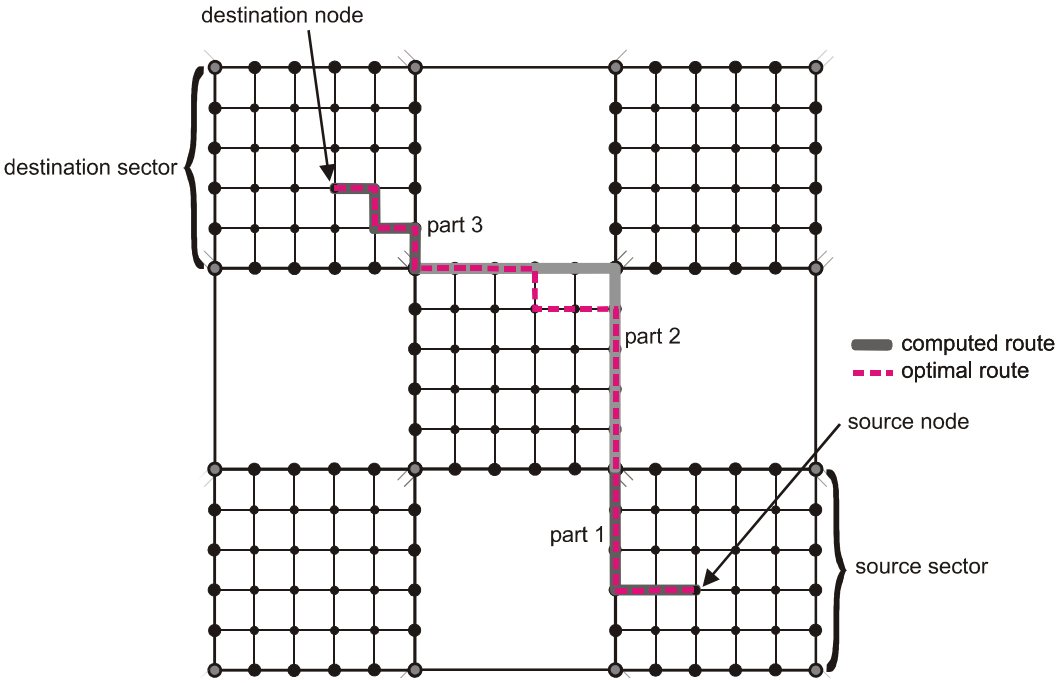

Figure 47: Route to a distant sector

Another suboptimal point can be the chosen strategy. The aim is to route the vehicles in shortest time from the source node to the destination sector. But to go from the source node in shortest time to the destination sector and from there in shortest time to the destination node is not always the shortest route. Figure 48 illustrates the fact.

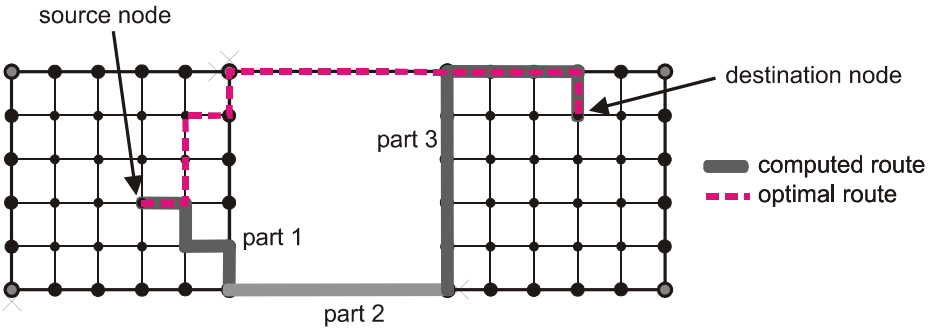

Figure 48: Shortest route to the destination sector

The computed route, consisting of the parts 1, 2 and 3, reaches the destination sector in the southwest because this is the shortest route in time to the destination sector. But from this point it is very far to the destination node, which is in the north of the sector. The optimal route goes along the northern motorway and reaches the destination sector in the northwest. The route to the motorway intersection in the northwest of the destination sector is longer than the computed route to the motorway intersection in the southwest. Nevertheless the route along the northern motorway to the destination node is altogether the shortest route.

To overcome this problem the strategy could be adapted. Instead of routing the vehicle in shortest time from the source node to the destination sector, the vehicles should be routed in shortest time from the source node to the destination node. This would mean that not only the most suitable motorway intersection of the source sector but also of the destination sector must be computed. So the shortest route would be the route from the 16 existing routes to the destination node, whose time is minimal, which is computed as sum from the time to reach the most suitable motorway intersection of the source sector from the source node, the time to reach the most suitable motorway intersection of the destination sector from the most suitable motorway intersection of the source sector and the time to reach the destination node from the most suitable motorway intersection of the destination sector. This strategy is more optimal but also much more complex. It results in much more communication necessary between the distributed Routing systems. And the routing tables of the Sector route finding systems of the motorway intersections must contain a time value in every row of the time column and time agents must be introduced to gather travel times for the routes from the motorway intersections of a city network to every node of the sector.
It must also be mentioned that the example network in Figure 48 constitutes a special, unrealistic case. In a real traffic network there are normally not two alternative motorways between two cities which besides have the same length. And the cities are not so close to each other. In a real traffic network a motorway goes to certain cities while other motorways go to other destinations in other directions. Here the strategy to reach the destination sector in shortest time is optimal in most of the cases because the travel times to a city along different motorways deviates strongly from each other. And the route along the motorway takes normally much more time than the last part of the route within the destination sector to the destination node. So the time to go from the destination sector to the destination node can be neglected.

6.1.2 Scalability

Scalability is one of the most important advantages of the Hierarchical routing system compared with centralized router. The chosen design of the Hierarchical routing system (see Figure 9 on page 15) allows the system to be distributed. Every Routing system can run on a separate computer system. By that the speed and available memory space increases.
By defining the sectors of a traffic network in an appropriate way, the size of the resulting city networks can be determined. In that way the responsibility of the Routing systems and by that the distribution of the Hierarchical routing system is influenced.
Also the Routing systems can be distributed among several computer systems to handle very complex networks. That is possible because the ABC-algorithm is a decentralized routing algorithm, what means that it does not presuppose that all data is available at one location. Therefore the motorway network (abstract level) and by that the Global routing system can be split up into several parts. And the same is possible for the city networks (detailed level) and the Sector routing systems. If networks are split up in such a way then the forward, backward and time agents must be able to go from one computer system, which is responsible for one part of a split network, to another computer system, which is responsible for another part of the same split network. This means that additional communication is needed what causes an additional delay. The extreme for the splitting would be to divide a network into so many parts that one computer system is responsible for exactly one node of

the network. This is theoretically possible but because of the resulting, enormous communication load between all the computer systems this could considerably slow down the performance of the system.
To sum up, scalability is ensured by the design of the Hierarchical routing system and the ABC-algorithm, but a good measure for the distribution must be found to ensure an optimal performance of the system.

6.1.3 Real time aspect

The real time aspect is very important for the system to be applicable in reality. Route requests of vehicles must be processed in real time. Vehicles need the computed route as soon as possible and only a short delay of at most a few seconds between the route request and the receipt is acceptable. That is especially important for driving vehicles which need the next steps of the route after the next intersection. If the next steps of the route are not delivered before the vehicle arrives at the intersection, the vehicle must stop to wait for the route. And this can be understood as failure of the Hierarchical routing system.
Besides, the optimization of the routes to adapt to new traffic conditions must happen in real time. If the adaptation to changed traffic conditions needs one hour this would mean a failure of the system. In that way not until one hour after a congestion occurred the vehicles are routed around it. But after one hour the congestion might be over and so the routing around the congestion is senseless then. For the adaptation of the routes to changed traffic situations at most a few minutes are acceptable. The faster the adaptation works the sooner the Hierarchical routing system can react to congestions or blocked roads.

The route computation modules of the Route finding systems (see Figure 19 on page 29) are responsible to receive route requests and to compute the shortest routes in time by using the routing tables. For routes that start in the network of a Routing system and end in the network of the same Routing system only the Route finding system of the Routing system is needed for the route computation. In that case the real time aspect can easily be ensured because the Route finding system has all information for the route computation available in its routing tables on the same computer system. Only the information in the routing tables must be read to determine the route.
Routes that start in the network of a Routing system and end in the network of another Routing system must be computed by two ore three Route finding systems like explained in section 5.4.2. Therefore communication between the concerned Route finding systems is necessary. The first part of the route is computed by the Route finding system of the network where the vehicle is actually in. After this Route finding system computed the first part it engages the next responsible Route finding system to compute the second part. And that Route finding system, if necessary, can again engage a third Route finding system to compute the third part. The second and the third part of the route are sent back to the first Route finding system which then sends the composed route back to the vehicle. In that case the route computation consists of two respectively three route requests to a Route finding system. Because a route request of a single Route finding system can be computed in real time, three route requests can also be processed in real time. The only question is how much is the communication delay between two Route finding systems. To request and return a route only a few hundred Bytes of data must be sent to a Route finding system. So the

communication delay is mainly dependent on the latency of the network and not on the bandwidth. For the internet the communication delay is normally only a few milliseconds. By that the communication delay is not a problem and so route requests that concern two or three Route finding systems can also be processed in real time.

The optimization of the routes and the adaptation to new traffic conditions is done by the route optimization module of the Route finding systems. These modules use the time estimates in the timetables and optimize the probabilities in the routing tables with the ABC-algorithm. The adaptation speed to changed traffic conditions depends mainly on two factors. First of all, the Timetable updating system must ensure that the time estimates in the timetable quickly adapt to the traffic situation. Secondly, the route optimization modules must adapt the routes to changed time estimates in the timetable as quick as possible. For the Routing system of [Kroon 2002] it was shown that for networks till a certain size the Routing system is able to work in real time and routes are adapted to changed traffic conditions after a few minutes. For larger networks more computational power (a faster computer system) is needed.
The Hierarchical routing system consists of several Routing systems which are each responsible for a part of the whole traffic network. Because the mode of operation of the Routing systems is in principle the same as of the Routing system of [Kroon 2002], the route optimization of the Hierarchical routing system is also expected to function in real time.

6.1.4 Robustness

The question of this section is: how robust is the Hierarchical routing system against the failure of parts of the system. The two main cases that must be considered are the failure of a Sector routing system and the failure of the Global routing system. With failure it is meant that the whole Routing system does not work. It is also possible that only the Timetable updating system or the Route finding system of a Routing system does not work. But if the Route finding system of a Routing system fails, then the Routing system is no more able to route vehicles and so the purpose of the system can not be fulfilled, even if the Timetable updating system still works. And if the Timetable updating system fails then the time estimates in the timetable will no more be updated and will not adapt to changed traffic conditions. Then the Route finding system is no more able to route vehicles by using dynamic data. Only static routing is possible what means a failure of the Route finding system because it is supposed to do dynamic routing. So the failure of a part of a Routing system means a failure of the whole Routing system. Therefore the failure of a Sector routing system or the Global routing system as a whole is considered here.

If a Sector routing system fails then the city network of the Sector routing system is affected. Vehicles inside the affected city network can no more be routed by the Hierarchical routing system. If vehicles inside the affected city need to be routed to a distant sector, then they will have to find the way to the motorway themselves. But when they arrive at the motorway the Global routing system will start to route them. For the other city networks and for the motorway network the failure of a Sector routing system has no consequences as long as the other Sector routing systems and the Global routing system has no malfunction. The routing in all other city networks and along the motorway network is

still working. Vehicles that are not inside the affected sector but that need to be routed to a street inside the sector can still be routed to the sector by the other Sector routing systems and the Global routing system. Only the last part of the route from the motorway intersection of the destination sector, where the vehicle arrives, to the destination node can not be computed. So when properly implemented, a failure of a Sector routing system affects only the corresponding city network and has no further consequences.

A major problem will occur if the Global routing system fails. First of all the routing from sectors to distant sectors along the motorway will not work any more. Secondly, the Sector routing systems receive no more updates for the time estimates of the surrounding motorway rings. By that the Sector route finding systems have no dynamic information about the state of the motorway rings. That means that also the routing in all city networks is affected by the failure of the Global routing system because no dynamic data for the motorway rings exist. A temporary solution to overcome with this problem could be to use default values as time estimates for the motorway ring. But then the travel times for the motorway rings are falsified and so the routing inside the sectors, which also includes the motorway rings, is no more optimal but nevertheless still working.
If at least the Timetable updating system of the Global routing system has no malfunction and the travel times for the motorway rings are updated at the Sector routing systems then the routing inside the city networks can be ensured.

To improve the robustness of the Hierarchical routing system, several enhancements can be done. An example could be to let the vehicles on a motorway send the update information in case of a failure of the Global routing system to the Sector timetable updating systems. Then the Sector timetable updating systems can compute the time estimates for the motorway rings themselves and are not dependent on the time estimates from the Global routing system. So the routing inside a city network is still working in an optimal way.
Another option is to split up the global network in parts and distribute the Global routing system among several computer systems. If properly implemented, the failure of parts of the Global routing system does not imply a total breakdown of the Global routing system. Missing time estimates can be replaced by default travel times for the links computed with the lengths of the link and the allowed speed. And for the parts of the Global routing system, which have a malfunction, static routes computed with Dijkstra's algorithm can be taken.

6.1.5 Applicability for realistic traffic networks

Till now the simplified network model, which was introduced in Chapter 2, was assumed for the Hierarchical routing system. This kind of network is at first glance far away from a realistic traffic network. All sectors are square, have the same number of nodes and links and are surrounded by a motorway ring. But this network is a very general network and can be changed into more realistic networks by easy adjustments like deleting nodes, deleting links, moving nodes and the corresponding links, splitting or combining sectors and changing the type of nodes.
Figure 49 shows a square traffic network with 9 city networks and surrounding motorway links which complies with the assumed network model. With the enumerated adjustments a more realistic network can be made out of this network.

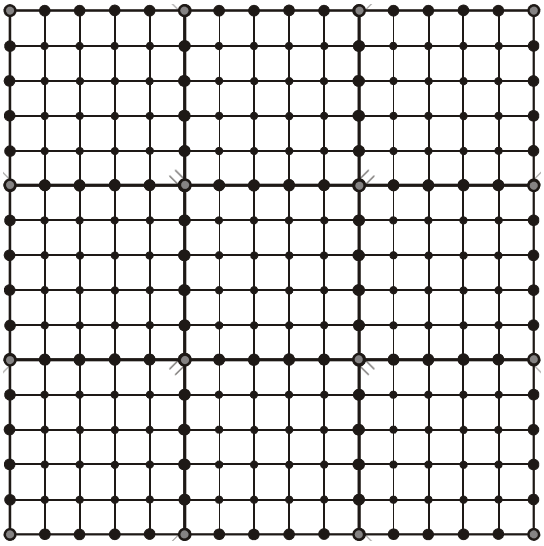

Figure 49: General network model

An example for a more realistic network is shown in Figure 50. This network still seems to be quite artificial, but with some fantasy it can be imagined to be a map of three real cities and motorways that connect the cities. In this example every of the three cities constitutes a sector. Only the city on the right is still surrounded by a motorway ring. The city at the top has a motorway connection in the south and a motorway that goes through the city. And the city at the bottom has a motorway connection in the north only.

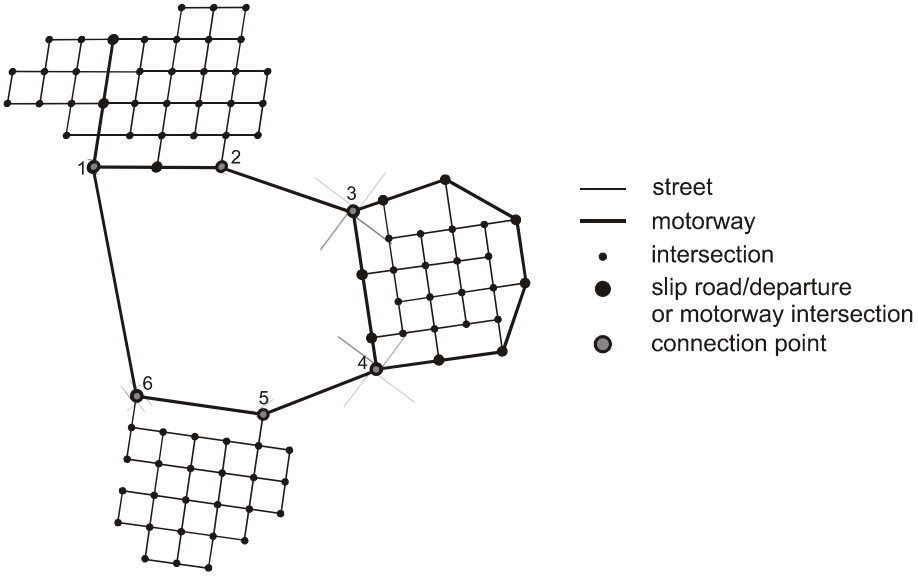

Figure 50: Traffic network with three cities

What has changed compared with the simple network model is that no more only the motorway intersections are the points along which a sector can be left. Also slip roads are nodes to leave a sector. In Figure 50 the nodes to leave a sector are marked as connection points. The connection points 1, 3 and 4 are motorway intersections and the connection points 2, 5 and 6 are slip roads/departures. Note that not every motorway intersection has to be a connection point.
For the Hierarchical routing system only very small adaptations of the algorithm are necessary to work with the adapted traffic network. Now the connection points play the role that the motorway intersections played in the simplified network. The connection points are the nodes to leave and to enter a sector. City networks can now have less, equal or more then four connection points. All the points that were valid for the motorway intersections are now valid for the connection points. The routing tables of the abstract level of the connection points now have time columns like the motorway intersections before. And the routing tables of the intersections (detailed level) contain time values for the travel times to reach the connection points. Instead of computing the most suitable motorway intersection the most suitable connection point is computed.
To sum up, it is only necessary to replace the motorway intersections by the connection points and no further changes are necessary. In that way the Hierarchical routing system is also applicable for other, more realistic networks.

Some examples for real city networks and the connection points are given now to explain how to define the connection points.
Figure 51 shows a city with a motorway or road in the north. The city including the three slip roads/departures of the motorway respectively road constitutes a sector. The slip roads/departures 1 and 3 are the connection points, the only points to leave the city. Slip road/departure 2 is not a connection point because to leave the city either node 1 or node 3 must be passed.

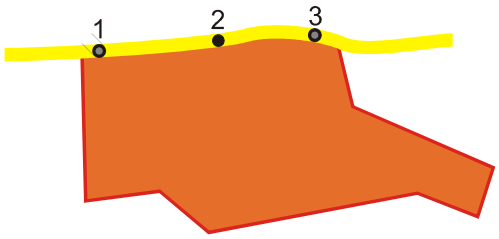

Figure 51: City with motorway along the north border

In Figure 52 a city with two motorways and a motorway intersection (node 3) is displayed. The connection points for that city are the nodes 1, 4 and 5. In this example the motorway intersection is not a connection point.

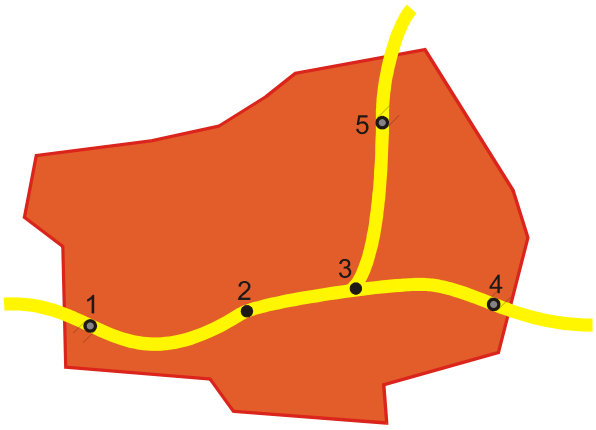

Figure 52: City with two motorways inside the city

Figure 53 shows a city where three different roads respectively motorways exist to leave the sector. But the roads/motorways do not go through the city. The connection points are the nodes 1, 2 and 3 where the roads respectively motorways start.

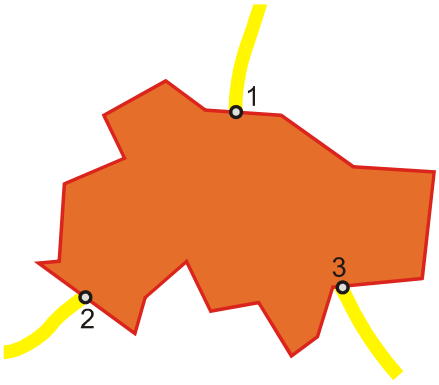

Figure 53: City with three connections

6.2 Recommendations

The focus of this thesis was concentrated on the development of a hierarchical routing system. It turned out to be much more complex then expected at the beginning of the project. Therefore not all ideas and possibilities to enhance the Hierarchical routing system could be considered in this thesis.
In this section the main points for further improvements are mentioned. They are aimed at the improvement of the route optimization by adapting the Timetable updating systems and the probability update formula of the ABC-algorithm which is used by the Route finding systems.

6.2.1 Timetable updating systems

The Timetable updating systems play a very important role for the route optimization. They provide the Route finding systems with travel time estimates for the links of the traffic network. Two factors are of prime importance for the Route finding systems. These are the goodness of the computed time estimates and the adaptation speed to new traffic conditions.
The only information source for the Timetable updating systems till now is the update information from vehicles. Because the time estimates are computed as average over all covered links the goodness is restricted. For all cars the same speed is assumed for the computation of the time estimates. But in reality different cars use different speeds. For example cars can drive a 120 at the motorways whereas lorries are not allowed to drive faster then 100 km/h. To improve this drawback the type of vehicle should be considered for the computation of the time estimates.
For streets, where vehicles regularly provide update information, a high adaptation speed to new traffic conditions can be ensured. But for congested streets nearly no update information is sent to the Timetable updating systems. That is because of the congestions vehicle need a very long time to cover the whole street. So the update information is sent after a very long time to the Timetable updating system and the time estimates are adapted to the congestion very late. For blocked roads where no car can drive any more this is even worse. No car at all provides update information about the blocked links because no car can pass the link. So the time estimates for those streets are totally inexact.
Several solutions to overcome these problems are possible. The vehicles could also provide their current speed to the Timetable updating systems and send update information after a certain time interval also if no street has been covered. Then if no street has been covered conclusions about congestions can be drawn from the speed of the vehicles. If the speed is very low then the time estimates for the link should be very high because it is very likely that the street is congested.
Also other sources can be used for providing dynamic data. These can be traffic information services or TMC. But the problem of those sources is that the given details about the traffic can not directly be converted into travel times. An appropriate way must be found to do it.
Another promising approach is the GSM technology. GSM can be used to determine the position of a mobile phone and by that the position of vehicles can be determined. Instead of the GPS satellites now the local GSM network can be used for tracking vehicles. Because a large part of the people own a mobile phone and also take it along with them when driving a car, the GSM network can also be used for another thing. From counting the number of vehicles that are tracked by GSM on a road conclusions can be drawn about the traffic density of the road. Roads respectively streets can be considered as buffer with a certain capacity. The capacity of the buffer corresponds to the maximum number of vehicles that have space on the road. If much free capacity is left in the buffer, what means a low traffic density, the travel speed of the road is not reduced. A quite full buffer means a high traffic density what results in slow moving traffic. And a full or nearly full buffer means that the road is totally congested and nearly no movement happens. In that way the state of the buffer can be used to compute time estimates for the corresponding road. But also sensors on streets can be used to determine the traffic density. Both ways are appropriate solutions to also get update information for congested roads and by that a fast adaptation for all streets and roads of the traffic network can be ensured.

6.2.2 Route finding systems

Besides the recommended improvements for the Timetable updating systems especially an improvement of the probability update formula of the ABC-algorithm can result in a great improvement of the route optimization. The update with the used update formula (see section 5.3.4) is very slow because it judges the quality of a route only by looking at the travel time. So the shortest route between very distant points, which still has a large travel time, receives a low probability increase only. But the route should receive a big update to be chosen as shortest route. Also in case of an congestion the probabilities are slowly adapted to the new traffic situation and so the adaptation of the shortest routes is slow. But especially in such situations the routes must be quickly optimized to react to congestions.
An improvement for the update formula to avoid the drawbacks is to store a history of paths found by the ants and the travel times of the paths. Newly found paths can be compared to formerly found path and the size of the update should depend on whether the new path is shorter or longer. By considering the differences between the last travel times congestions can be detected. In case of a suddenly occurring congestion the travel times strongly go up what means a high value of the first derivative of the travel times. Then the probability increase ΔP can be increased by increasing the value of the constant A.
Another way to detect congestions are the methods which were introduced in the last section. GSM or sensors on the roads can be used to determine congestions. With that information the constant A can be increased for congested road to ensure a fast adaptation to changed traffic conditions.

Appendix: Bibliography

[ANWB] ANWB, www.anwb.nl

[Dorigo 1999] Dorigo M., *Artificial Life: The Swarm Intelligence Approach*, Tutorial TD1, Congress on Evolutionary Computing, Washington, DC, 1999.

[Eggenkamp 2001] Eggenkamp G., *Dynamic Multi Modal Route Planning: An Artificial Intelligence Approach*, Master's thesis, Delft University of Technology, 2001.

[Engelbrecht 2002] Engelbrecht A. P., *Computational Intelligence, An Introduction*, Wiley, 2002.

[Horowitz 1998] Horowitz E., Sahni S., Rajasekaran S., *Computer Algorithms*, Computer Science Press, 1998.

[Kroon 2002] Kroon R., *Dynamic vehicle routing using Ant Based Control*, Master`s thesis, Delft University of Technology, 2002.

[Russell 1995] Russell S., Norvig P., *Artificial Intelligence, A Modern Approach*, Prentice Hall, 1995.

[Siemens] Siemens, www.siemens.de

[Tatomir 2002] Tatomir B., *Real Time Simulator for Ant Based Control Algorithms*, Diploma thesis, Delft University of Technology, 2002.

[TMCForum] TMCForum, www.tmcforum.com