Technische Universiteit Delft

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Man Machine Interaction Group

Dynamic Routing using Ant-Based Control

by

Adriana Camelia Suson

Supervisor: Prof. Dr. Drs. L. J. M. Rothkrantz

Delft, August 2010

Dynamic Routing using Ant-Based Control

Thesis

submitted in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE
in
MEDIA AND KNOWLEDGE ENGINEERING
by

Adriana Camelia Suson
born in Braşov, Romania

Media and Knowledge Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
Delft University of Technology

Dynamic Routing using Ant-Based Control

by Adriana Camelia Suson

Department: Man Machine Interaction `http://mmi.tudelft.nl/`
Student Number: 1391968

Date: 23 August 2010

Committee Members:

Member: Prof. Dr. Drs. L. J. M. Rothkrantz, TUDelft

Member: Dr. Ir. P. Wiggers, TUDelft

Member: Ir. H. Geers, TUDelft

Member: Bsc. B.Tatomir, TUDelft

Abstract

Currently most car drivers use static routing algorithms based on the shortest distance between start and end position. But the shortest route is different from the fastest route in time. Because existing routing algorithms lack the ability to react to dynamic changes in the road network, drivers are not optimally routed.

In this thesis we present a multi-agent approach for routing vehicle drivers using historically-based traffic information. The general workings of our solution bears strong similarities with Ant Based Control (ABC) and AntNet, but an important modification has been made, namely the adaptation of ant-like agents for spatio-temporal routing.

The dynamic routing algorithm proposed, routes self-interested drivers on an intersection to intersection basis via the fastest path between a proposed source and a destination. For this to happen, a time-expanded graph encodes variable road network costs. Ant-like agents are launched in this graph. They use a technique of collective learning based on locally dependent pheromone tables.

Finally, we report results obtained for part of The Netherlands' GIS-based road network. In the established experiment setting, the new ABC makes a positive difference for drivers. An important reduction of the travelling time was observed in 53% of the cases. The experimental results also showed that ABC clearly outperforms Static Dijkstra's algorithm and Dynamic Dijkstra with updates algorithm.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

Symbol	Description
$G = (V, E, L)$	denotes a directed graph of a spatial network model
$V = \{1, \dots N\}$	set of nodes/vertices/intersections/junctions
$E = \{e_{ij} \mid e_{ij} = (i, j),\ i, j \in V\}$ and $\lvert E\rvert < \frac{n(n-1)}{2}$	set of directed edges/links
$L_i = \{l_1, \dots, l_r\}$	set of locations for intersection i
$L = \{L_1, \dots, L_i, \dots L_N \mid 1 \leq i \leq N\}$	set of all locations in the spatial network model
N	number of nodes/vertices/intersections in V
M	number of edges in E
$R = (e_{i_1 i_2}, e_{i_2 i_3} \dots, e_{i_{p-1} i_p})$, $i_1, \dots, i_p \in V$	a route as a sequence of consecutive edges
$\mathcal{I}_k = [start_k, end_k]$	discrete time interval beginning at time instant $start_k$ and ending at time instant end_k
$\mathcal{T}$	time horizon
$\mathcal{I}_\mathcal{T} = \{\mathcal{I}_0, \dots, \mathcal{I}_T\}$	set of time intervals for $\mathcal{T}$
$c^r_{ij(t=to)}$	cost of route r between node i and j starting at time instant t
$d(i, j)$	distance of the edge connecting node i and node j
$t_{ij}(\mathcal{I}_k)$	estimated traveling time from node i to node j, $i, j \in V$, when starting in the interval $\mathcal{I}_k$ at node i
t_i	departure time from node i
t_{ij}	instantaneous traveling time from node i to node j, $i, j \in V$
$\mu_{id}(\mathcal{I}_k)$	average travel time from node i to destination d when starting at node i in the interval $\mathcal{I}_k$
$v_{ij}(\mathcal{I}_k)$	actual average speed when travelling between i and j in the time interval $\mathcal{I}_k$, $i, j \in V$
v_{ij}	instantaneous average speed on the segment (i, j), $i, j \in V$
$v_{ij}(t)$	free flow travel speed from node i to node j at time t
c_{ij}	capacity of the segment (i, j)
R_i	routing table of node i
$\mathcal{F}_{sd}$	forward ant going from source s to destination d
$\mathcal{B}_{ds}$	backward ant, the corespondent of the forward ant $\mathcal{F}_{sd}$
$\mathcal{P}_{ij}(\mathcal{I}_k)$	pheromone level of the edge connecting node i and node j in the time interval $\mathcal{I}_k$
R_i	the routing table of node i

Symbol	Description
$\mathcal{S}_{sd}(k)$	stack memory[1] of the k-th forward ant $\mathcal{F}_(sd)$ (s source node, d destination node)
$(i, t_{si}),\ s, i \in V$	element of the stack $\mathcal{S}_{sd}(k)$ is a set of pairs of the form: node i and travelling time t_{si} to the respective node if staring in s.
$\mathcal{N}_i$	set of nodes connected with node i
r	reinforcement parameter used to update routing tables

1. A stack is a last in, first out (LIFO) abstract data type and data structure. The stack is characterised by only two fundamental operations: push and pop. stack grows upwards from its origin. The stack pointer points to the current topmost cell on the stack. A push operation increments the pointer and adds data to the top of the list, hiding any items already on the stack, or initialising the stack if it is empty. The pop operation removes an item from the top of the list, returns this value to the caller and decrements the pointer. A pop either reveals previously concealed items, or results in an empty list.

Acknowledgements

In the first place, it is not only the ants that have to be "held responsible" for the direction adopted in this thesis. There are also a few very special people that have contributed in some way or another to this thesis. I would like to say thank you to that people that I have lived around and I have met during the last few years. They have been the closest ones, either from a scientific, from a technical point of view and/or from a personal point view.

Upon finishing this thesis project, my deepest appreciation goes to the chief of the committee Prof. Dr. Drs. Léon J. M. Rothkrantz. His efforts and advices have always helped and guided me through my degree pursuit. This thesis could not have been completed without his insight and supportive supervision.

Additionally, special thanks go to Bogdan Tatomir, for introducing me to the Quintiq Environment, for obtaining software licenses for this environment and for his all time scientific and non-scientific support.

I am grateful to my family for all the warm words and the encouragements they have offered me during all these years that passed. I really enjoyed my stay in the MMI lab and I thank all the colleagues that I had there, for the relaxing moments spent together. In particular, special thanks also to Ruud and Bart, for technical support whenever I was confronted with laptop failures and not only that. Not to forget, I would also like to thank Joe for everything that has done and is still doing for me.

Thinking back, I realise that maybe there are still other people that I should thank. To all that were not mentioned here but they have, consciously or unconsciously taken part to my own growth: Thank you!

Adriana Camelia Suson,
23 August, 2010
Delft.

Chapter 1

Introduction

1.1 Problem setting

In our contemporary times everything seems to be related and interdependent in a certain proportion of mobility. Since human society is spread over different regions of the globe, the access time provided by transportation is indispensable for economic and social development. Transportation systems are believed to have nurtured economic growth by facilitating trade and permitting access to resources. Transportation has also expanded the possibilities of social and cultural connections, increased development and favoured consumption.

Usually, the demand for mobility increases proportionally to the population growth rate. In densely populated regions the capacity of roads is not sufficient to optimally route all drivers from a source point to a destination point. In other areas traffic is reaching gridlock. During the last years the burden caused by growth of traffic intensity and frequency of traffic jams on major roads, highways and in urban areas has been stigmatised.

However, regrettably enough, the expansion of infrastructure cannot always keep up with these increased demands so other solutions have to be taken into account to solve traffic problems. We simply cannot build enough roads to cope with the traffic increase. Building wider motorways is not seen as the way forward as it would attract yet more traffic and therefore would not act towards stopping or reducing the congestion problem. Unless we can deal with congestion we will all have to face the consequences.

Congestion causes loss of time and money, it damages the environment, encourages energy waste and it affects our health. In fact we have to think about travelling in a completely different way. We need to make use of the already existing road networks and we need to make smart choices about the way in which we travel. It is our duty to think about how we can make journeys more time-efficient and free of congestion.

In most developed countries there are research programs, therefore, which deal with the causes and effects of congestion. Mechanisms for congestion prediction have been considered and countermeasures affecting *road infrastructure* (e.g. reversible lanes, improving network junctions, separate lanes for specific user groups), *supply and demand traffic management* (e.g. adding more capacity at bottlenecks, road pricing, road space rationing) as well as *intelligent transportation systems facilities* (e.g.traffic reporting, variable message signs, navigation systems, automated highway systems etc.) are being applied. These are of limited help as they offer no alternative route when traffic jams are formed instantaneously.

A traffic-jam is a dynamic phenomenon in which every motorist is slowed down by his predecessor, Figure 1.1. In The Netherlands, during every workday there are more than two million Dutch drivers caught in traffic-jam. At some time every road user is

Figure 1.1: Traffic congestion in The Netherlands on A9 at the off-ramp towards N205.

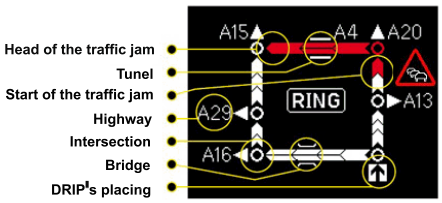


Figure 1.2: Dynamic Routing Information Panel (DRIP) on the ring of Rotterdam.

Graphical information is displayed.

Figure 1.3: Minutes of delay on a DRIP for drivers heading towards Amstel intersection.

Only text information is provided.

going to be stuck in a traffic-jam. To address the issue, Dutch road users are informed mainly on highways about the current movement conditions using DRIP (Dynamic Route Information Panel). DRIPs are programmable graphic and textual panels, where symbols and/or text can be displayed. These panels allow the user to have better access at traffic information in a graphical form (see Figure 1.2). Route information such as travel times (in minutes) or traffic-jam lengths (in kilometres) for a specific route can also be provided (Figure 1.3). Besides, there is the alternative of warning users about incidents, about upcoming congestion and about certain weather condition like fog or ice. Through these panels information traffic managers attempt to improve road conditions. The display of congestion lengths and duration on maps is also quite popular in The Netherlands (see Figure 1.4). For instance, two of the most visited websites in the Netherlands that display traffic information to the public so that travellers can make an informed choice about their trip routes, are `http://www.traphic.nl` and `http://www.anwb.nl/verkeer`.

Surveys have indicated that motorists' main concern is the uncertainty of their journey times. Because of that, several automotive navigation manufacturers have developed and advertised navigation systems for congestion avoidance. These devices have an integrated real-time traffic and road incident receiver to display information based on Traffic Message Chanel (TMC), RDC or GPRS/3G. For example, the navigation manufacturer **Tom Tom** (Figure 1.5) provides real-time traffic information based on cellular floating phone data system which exploits data from anonymous cell phone users and is enhanced by a GPS-based probe information and other third party messages. A similar application is the **Mobile Millennium Project** (Figure 1.6) in Berkeley California, which uses anonymous speed and position information gathered by GPS-equipped mobile phones, fuses it with data from static traffic sensors and broadcasts the traffic information back

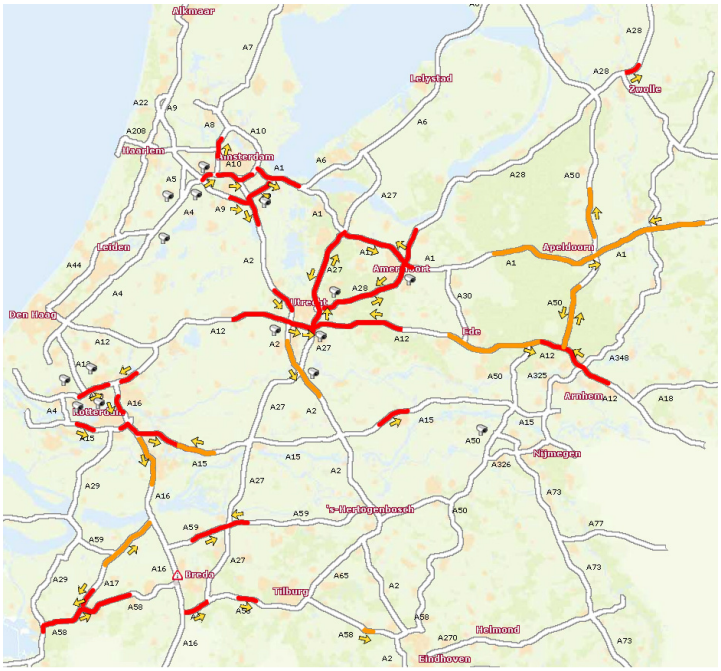


Figure 1.4: Map display of traffic congestion in The Netherlands.

This situation was recorded on `www.traphic.nl` on 10 February 2010 at 18:42 during a snow fall. The red segments indicate places where congestion has occurred.

Figure 1.5: Portable GPS car navigation system of the Dutch manufacturer Tom Tom.

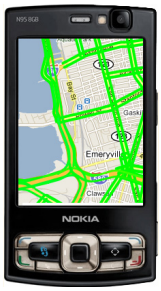


Figure 1.6: Mobile Millenium traffic-monitoring system based on GPS cellular phones.

to the phones. In the case of Tom Tom, the measurements are compared with historic speeds called *common "ground truth"*. The deviation against the ground truth is represented by a reliability measure in a number of parameters like: speed, delay in reporting a jam, changes in a jam, or positive and false traffic jam measurements when the ground truth indicates a traffic jam. The reliability indicates the confidence in the speed measurement. The most reliable data source carries more weight. Besides the reliability, age of the measurement is considered together with other parameters. A data fusion engine is responsible for delivering the resulting information. The update frequency of traffic information is 3 minutes. The information computed through this concept is far more accurate and precise compared to the TMC message. Furthermore, the delay time and the reporting of the delay are more realistic assuming that the GPS probe can be seen as a ground truth. However, as this navigation manufacturer also points out, for sections

Table 1.1: List of the top 10 *filezwaarte* days in The Netherlands.
Information from `InfoNu.nl`.

Nr.	Date	Length ($kmmin$)	Main Causes
1.	8 February 1999	975	snow fall, rolled over freight carriages
2.	25 March 2008	888	snow, day after Easter
3.	25 November 2005	810	snow, evening rush-hour
4.	26 February 2004	760	winter weather with snow fall
5.	17 December 2009	671	snow fall, morning pip
6.	6 March 2006	660	snow and end holiday, morning pip
7.	26 May 2009	605	thunderstorms, morning pip
8.	13 November 2006	594	rain, morning pip
9.	16 November 2006	592	rain, evening rush-hour
10.	29 January 2004	571	snow fall and heavy wind, morning pip

along a route that are far ahead in space from the current vehicle position, the use of real-time traffic data doesn't make much sense since the travel time may change before the driver has reached these road sections. So far, time-dependant speed profiles for the network graph are good for a first approximation if predictive data is not available. More detailed representation of Tom Tom's concept of time-dynamic navigation using dynamic, historic and travel-time information gathered from GPS and GSM probes can be found in [67].

Another issue that has been identified by the researchers with respect to commercial navigation systems is that static navigators offer as advice every time the same routes as shortest detours, in spite of offering an alternative personalised for a specific driver. This kind of method determines the occurrence of heavier congestion at the network level. Moreover, we need to accentuate that dynamic routing will be more beneficial for the drivers.

In The Netherlands, traffic congestion management relies heavily on traffic management. But in most cases traffic management is used only on a local level and it lacks integrated and network wide approach. Recently there was also an initiative to establish an operational **National Data Warehouse (NDW) for Traffic Information** on a basic network of at least 5,500 kilometres of national, provincial and municipal roads within 4 years. In 2006 almost 1000 kilometres from 2350 of highways were monitored. The measurement were then converted in data per minute and sent to regional traffic control centres. NDW will become the databank that will collect, process, store and distribute all relevant traffic data. The available information that is going to be stored is: velocity on trajectories, density, spotspeed and event information on traffic. The system will also benefit from location referencing systems based on (x, y) point coordinates and from TMC. However, this helps very little in facing up to the congestion frustration.

For the purpose of illustration Table 1.1 lists the top 10 busiest days in the history of traffic in The Netherlands, according to the ANWB organisation. *Filezwaarte* or traffic heaviness is the length of the traffic-jam multiplied with the duration. File gravity is expressed in kilometre minutes: the number of minutes that a particular file, with a certain length, existed. A traffic-jam of 3 km for 5 minutes and afterwards 2 km for 3 minutes produce therefore a traffic-jam heaviness of $(5 \times 3) + (3 \times 2) = 21$ kilometre minutes (km min).

A major drawback is due to the fact that today's traffic information providers rely almost completely on real-time data and historic information. Although maybe some years ago, we would have been more than satisfied with this kind of data, the situation nowadays is different. The experience of many motorists has shown that the so called "real-time" information broadcasted via navigation systems or via Variable Message Signs

may not always turn out helpful. There are two situations that need to be acknowledged in support of this claim. First we need to consider that an individual traveller needs a non-negligible amount of time to travel across the network and the network state can vary during this time. Secondly, a trip is planned in advance in most cases, when no information about future traffic congestion is available. For that purpose, **Inrix** [53] offers **dynamic predictive traffic information** in the United States, based on modelling technologies developed by Microsoft Research. Flow patterns can be predicted in 15 minutes intervals for a time horizon of up to a year into the future. However, Inrix is more a planning system for travellers than a navigation system which offers the precise list of turns and links that have to be followed to reach the desired destination.

Subsequently, there is great interest not only in using real-time traffic information but also in using predictive traffic information to intelligently route, optimise departure time, avoid incidents and to improve mobility in general. But, there are still gaps in data collection, data fusion and reliability of data. The ultimate goal is perceived recently as integrated regional multi-modal planing based on a system-wide real-time multi-modal data collection, integration and analysis. Besides, it is desirable network optimisation in response to incidents.

Several **variables** should be studied when offering routing advice. Some drivers may be more flexible than others. Consider, for instance, the return journey from work to home, when drivers can select the departure time. If no reliable traffic information is available, a flexible driver might not be able to avoid the peak traffic conditions, and in this way will increase both his individual journey time and the uncertainty for other individuals. But, whenever a driver avoids traffic congestion it makes an easier job for those who do not have the flexibility of choosing the departure time.

It is extremely important to consider the actual travel behaviour and the implications of choosing a particular route. Among other things, user preferences sometimes need to be taken into account when prescribing a route from source to destination. Some drivers will prefer to save time, others to save fuel. Some users will accept to be guided using routes through cities, others will strictly prefer the highway. Making stops or meeting a certain time window can also be crucial. Last but not least, departure time is an important element when when embarking on a trip because naturally, the motorists have the possibility of choosing the departure moment. By considering all these variables, a multi-objective decision situation arises.

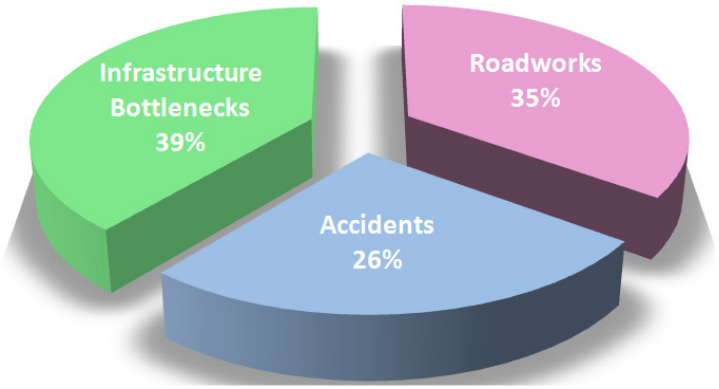


Figure 1.7: Causes of travel time loss on German highways, adapted from [54].

Usually, a top question coming from motorists is whether alternative roads are worth following. When there is a congestion ahead on the motorway should drivers stick with it or find an alternative route and get off the motorway? Here, people need to know the difference between **recurring** and **non-recurring** (incident) congestion. Recurring

congestion is attributed to insufficient capacity and ineffective management of traffic capacity, while the non-recurrent congestion appears under unexpected circumstances such as car accidents, working zones, bad weather, special events and emergencies. As figures show, in Germany the non recurrent congestion, in the form of roadwork and incidents, accounts for 61 percent of travel time of losses on German motorways (see Figure 1.7). In the first case, when congestion is recurrent, it is advisable to stick to the motorway because the other alternative routes are already invaded by traffic, commuters have already figured the best other choices. Even if the alternative is less congested it is always longer, because rational drivers choose the shortest routes first. But the situation is different when an incident happens, where no one would have expected. Then, it is recommended to get off the motorway since the alternatives will be opened. Moreover, it is believed that traffic information can have the greatest effect in non-recurrent traffic conditions.

In traffic modelling field there is a distinction between **System Optimal (SO)** and **User Equilibrium (UE)** *flow*. These notions go back to Wardrop [69]. The same difference between SO and UE can be found in *routing*. The goal is either: routing for best possible network performance (SO routing) or route optimisation of many selfish users with conflicting interests. The later is often criticised because the result is not globally optimal and the cost is the decrease of network performance.

In a SO situation, the total travel time of the whole network is minimised. But the approaches that support SO may discriminate some users in favour if others. On the other hand, in a UE situation each driver acts selfishly and chooses the route with the smallest possible cost function, which suits his own interests. Equilibrium is achieved when every trip-maker is using the best route possible, given prevailing congestion levels. This is an equilibrium since no user can switch routes and improve his/her travel time and so no user will switch voluntarily. Selfish drivers create *"selfish flows"* [3]. All the aggregated traveller's choices will determine how the traffic spreads over the network.

As far as the optimal travel time in a network is concerned, the **traffic assignment problem** has been elaborated. Traffic assignment's main objective is to determine the optimal trade-off between the infrastructure supply and traffic demand so that travellers choose their optimal route. But in this case, the travel demand is given as an origin-destination (OD) matrix with the number of trips between origin nodes and destination nodes. Moreover, the assignment problem can be either static or dynamic, depending on how many time periods are considered. Static assignment is however based on unrealistic assumptions (that a vehicle will contribute to all links along the route at the same time, since only one time period is considered) and is believed to be a deprecated method.

1.2 Problem statement

1.2.1 Purpose

Due to its wide impact in the transportation field, routing has received a lot of attention from the academic world. But routing is not only a well known optimisation problem intensively studied by the scientific field; it is mainly a day-to-day need of individual travellers. It goes without saying that every road traveller desires a smooth journey, with as little incidents and as few traffic congestion as possible. Route guidance is supposed to offer support and give advice for reducing the mobility related issues.

The previous years have increased the focus on effective routing. Nevertheless, there are still several important matters in the field of routing algorithms that have not completely solved. Routing community efforts have been previously dedicated to overall robustness and single shortest path solution in static environments. Yet, most of the real world conditions in which vehicle traffic routing takes place are not completely static. Overall, congestion has a highly dynamic character for forming and dissolving and no

static approach, applied either globally or individually, is satisfactory enough to avoid it [72].

As soon as traffic flow conditions change routing strategies have to be changed to address new situations. As a result of the recent advances in technology and in communication technologies, real-time traffic routing has been considered one of the most promising approaches to relieve congestion's effects. In this approach sensors like inductive loops are used to continuously gather and monitor the traffic flow and then a routing strategy based on this information prescribes routes to drivers. Yet, using real-time information is computationally difficult.

The problem we want to confront in the present thesis is the need for proper routing, so that capacity of all existing roads is exploited in a more optimal way. The best/optimal route can be defined in terms of several quality metrics and the possibilities should be carefully considered. Both the perspective of the user and of system equilibrium should to be taken into account. Simplicity is also a desirable characteristic for routing algorithms because minimum software resources should accomplish the task. But above all, we are interested in algorithm's ability to perform in case of high traffic conditions and during incidents, a property called robustness. Robustness is related to the flexibility or adaptability to a variety of conditions. Nonetheless, scalability, or the capacity to operate in large networks without an increase of resources is desired. The path selection must be *distributed* and should take into account the evolution of the state of the network which is influenced by the traffic flow patterns.

1.2.2 Statement of objectives

In general terms, our main focus is to design a flexible, and, at the same time, robust traffic routing algorithm that will support self-interested agents in the context of dynamic network traffic conditions. We plan to exploit as much as possible the benefits of swarm intelligence algorithms for the purpose of routing vehicles and helping road users to navigate in networks easier. Moreover it is also desired that a time (delay) aware routing is performed, taking into account the current traffic situation and maybe the future expected traffic movement. Last, but not least, the routing advice offered by the algorithm should return the fastest route in time.

The main types of objectives that we set are the ones below:

1. ***Related work review.*** *Obtaining necessary theoretical knowledge concerning routing.*
2. ***Algorithm design or adaptation.*** *The design or adaptation of an algorithm based on ABC, with focus on adaptive/dynamic routing with respect to traffic variation, for drivers in a road network, in the conditions of dynamic traffic flow. Routing should be performed for highways and national roads on an intersection to intersection basis with the purpose of avoiding congestion. Routing should consider the current traffic situation and maybe avoid future traffic congestion of the road network.*
3. ***Implementation of a running prototype*** *that exploits existing traffic information sources.*
4. ***Analysis and Validation.*** *Analysis of design choices and validation of the soundness and robustness of the proposed system. Analysis of the competitiveness with static/dynamic routing algorithms. The understanding of the workings of the algorithm.*

Table 1.2 describes which actions are performed in order to reach the proposed goals.

Table 1.2: Points of the research methodology. Actions and goals of our research.

Goal	Action
Literature review.	• The review of related issues to adaptive routing in dynamic context. • Describing the current state-of-the-art routing algorithms with focus on swarm intelligence. • Analysing the advantages and disadvantages of current routing techniques.
Algorithm design or adaptation.	• Design a general model of the routing system that incorporates available resources like a spatial network model and traffic information. • Revise how path costs are computed. • Modify an adaptive swarm intelligence algorithm so that it integrates present and future realistic route costs.
Implementation of a running prototype.	• Describe the chosen architecture of the prototype. • Analyze the UML model of the implemented routing system. • Report the main pseudo code parts of the routing algorithm implemented. • Present the Graphical User Interface (GUI).
Analysis and Validation.	• Design and implement a time flow simulator. • Design and implement a graphical interface showing moving vehicles. • Compare the proposed algorithm with a static and a dynamic routing algorithm. • Conduct experiments to determine the functioning, validity and robustness of the routing system.

1.2.3 Scope

Developing an effective routing mechanism is multifaceted task. The complexity of traffic routing and management is related to the interaction of all travellers with their preferences, vehicles, road conditions, and special events. The dynamic essence of traffic is perceptible in spatial and temporal changes of traffic demand.

We limit, however, the scope of our research efforts as follows:

- First of all, only motorised vehicles such as cars, trucks, vans are considered for routing. There are no other traffic participants besides vehicles. Bicycle and pedestrian routing is beyond the scope of this thesis.
- Secondly, we will consider motorway uninterrupted traffic flows. No traffic lights will be taken into account. The routing that we mention here is going to be interurban routing.
- Tolls for using certain highways and toll gates are not considered either.
- Additionally, out of the scope of this thesis is modeling driver's sum of preferences or behaviour in response to the routing advice offered.
- Routing often involves man-machine interactions (interactions between driver and hand-held device or User Interface), but we don't focus on this particular aspect. The user experience is out of the scope of this thesis.
- Likewise, the occurrence and causes of particular incidents and events that affect traffic conditions are not going to be discussed. We will not model incidents that occur in traffic due to bad weather conditions, roadway maintenance, roadway design, equipment failure or driver behaviour.
- Our routing model doesn't consider that certain drivers drive slower/faster than other drivers on the same road. This is another limitation of our model.
- Moreover, it should be considered that drivers do not end their trip on a road, or in an intersection. Drivers may also stop along the route, but this out of the scope of this thesis.

1.3 Societal relevance of the thesis

The rush hour brings saturation of traffic on numerous highways. Ironically, more traffic will signal a better economy. But above all, the traffic jams also induce costs to society at several levels: economically and socially.

Delays caused by congestion often result in unproductive hours for commuters and influences the economies of regions where it happens. Nonetheless, getting caught in traffic forces people to change their travel behaviour (e.g. by departing earlier or choosing alternative mode of transportation). Transportation is an experience shared by many people, and directly affects their well-being. Congestion acts mainly as a stress factor, encouraging road rage of frustrated motorists. Traffic jams result in an increase of fuel consumption which in turn raises the level of air pollution. Since in congested areas like Randstad, in The Netherlands, the infrastructure can hardly be expanded, improving the traffic management and the routing advice given to drivers are always alternatives that deserve to be tried.

Given the huge costs of congestion for society and the urgent needs for a solution, we investigate in this thesis the effects on incorporating actual travel time information into a routing algorithm so that current and future congestions could be avoided by the drivers. The work presented in this thesis can contribute to the solution of congestion in the following way:

- Dynamic rerouting and the provision of reliable traffic information are part of the component of a traffic management system. By providing users with alternatives, the impact of non recurrent congestion is reduced.
- The wasted time of motorists and passengers will be reduced, which will favour regional economic health.

- Having a navigation system that is based on an adaptive routing algorithm, will reduce the burden of not knowing what is the exact arrival time at the destination. Efficient traffic routing may diminish the stress level of travelers, especially in the life of the frequent commuter.

Nevertheless, we acknowledge that a good route planning algorithm is not the only direction that has to be adopted so that congestion is reduced.

1.4 Research challenge

Design of an adaptive and optimised vehicle routing strategy

So far we have talked about the context in which this project takes place and we have stated the objective and the scope of this thesis. We pointed out the importance of mobility and how this is negatively influenced by congestion. The importance of implementing effective routing systems is, therefore, apparent. Today's routing systems are complex, however, an important shortcoming can be spotted. Traffic variations are left aside or they don't receive enough consideration in the current routing systems for vehicles. Moreover, they usually forward vehicles for the same destination over the same path. This results in algorithms that are robust and have rather predictive behaviours, but that are not really adaptive.

The main challenge in the case of our problem is designing an algorithm that has the ability to adapt to the global changing vehicular traffic patterns. The algorithm should integrate measured or reported traffic delays and produce an optimised path for every route request. The algorithm should offer the fastest path in time. Because traffic delay is varying during a day, the path used for forwarding vehicles will change dynamically over time intervals.

1.5 Organisation of the thesis document

In this section, we present a general overview of the topics discussed in the present thesis together with their organisation in chapters. A schematic view of the thesis' chapters is reported in Figure 1.8.

The problem context, the research relevance, the problem statement and the general objectives implied by the current project were presented in the introductory chapter. Chapter 2 gives a synopsis of the related work that has been considered influential to the subject of the research. In the first part, the most important classes of route guidance systems are concisely presented. The second subsection of Chapter 2 is concerned with Dijkstra's algorithm. The third subsection of Chapter 2 inspects algorithms from state-of-the-art approach to routing, namely from Swarm Intelligence. Biological inspiration is acknowledged for static and dynamic routing, for vehicle traffic routing and for telecommunication routing as a similar problem. Several hierarchical nature-inspired algorithms are also compared as response to scalability issues generated by previous algorithms in large routing networks. It is unrealistic to consider that traffic problems can be solved altogether by traffic routing. Therefore, it is important to get a good understanding of how traffic queueing models can be beneficial in the context. There is also a short section referring to this problem. The second chapter ends with a conclusion on the main techniques available in the traffic routing domain and points out a gap that must be filled in.

After stating the existing related work, it is the moment to give an insight into our research focus. More specifically, general design aspects, the required resources for the routing system and a description of the algorithmic approach constitute the subject of Chapter 3. Based on the previous literature study we describe a new algorithm, an extended version of Ant Based Control and AntNet, coming from the field of Swarm

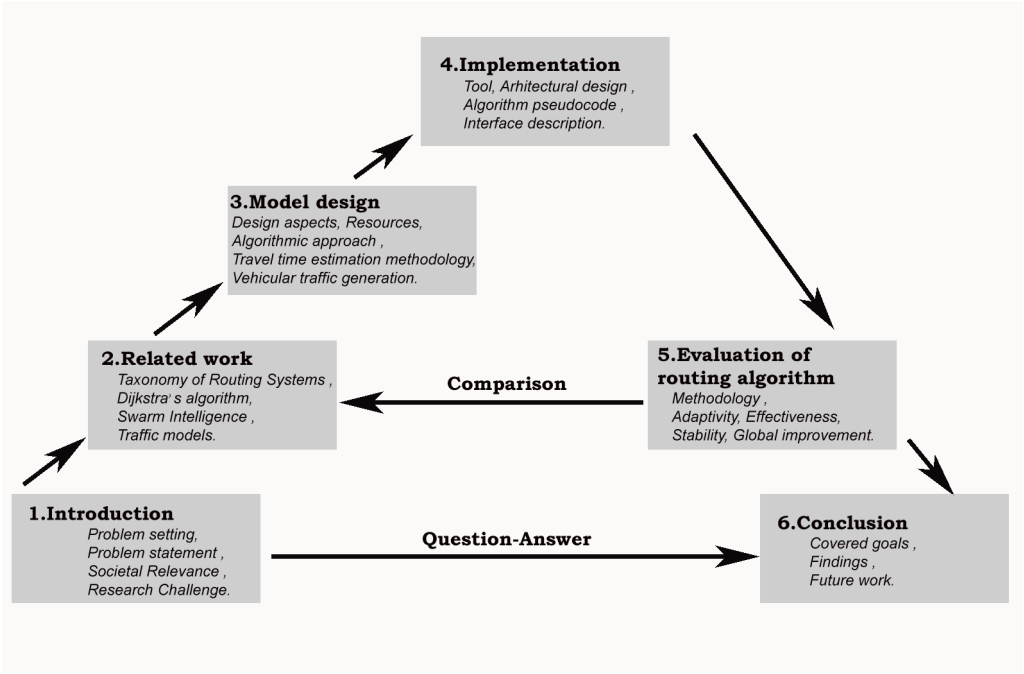


Figure 1.8: Schematic view of thesis organisation.

Arrows represent connections between chapters. The text in Italics reflects the main topics related to the chapter.

Intelligence. We add a formulation of the travel time estimation method and discuss how vehicular traffic data is integrated.

The topics considered in Chapter 4 are related to implementation details. This chapter also gives a brief overview of the development tool (Quintiq Environment). System's architectural design, the algorithm pseudo code and the description of the interface represent the important sections of this chapter.

Chapter 4 reveals the evaluation methodology, including experimental settings and data processing. The response of the algorithm to real situations gives a good insight into the performance of the algorithm. Analysis and discussion is performed for several experimental settings.

Finally, the thesis ends with our conclusion that consists of the reached goals and the accumulated findings. We hope that by combining the knowledge accumulated in this research we covered in a sufficient manner the proposed goals. Directions for further work and possible extensions are also integrated in this chapter.

Chapter 2

Related work

In the lines to follow we present a high-level overview of the most important steps in the process of developing a route guidance algorithms, aiming to focus on the previous research and the previously mentioned solutions. In order to solve the **routing problem**, a number of approaches have been proposed and several types of systems have been developed. The early approaches include static shortest-path routing, e.g. Dijkstra's algorithm, but they have the disadvantage of lacking adaptivity to network conditions. The desire is however, to have a routing algorithm based on the observation of network conditions, designed to generate routing advice in accordance with predicted flows in the network. The literature in the domain of routing is quite extensive, but we will restrict ourselves to what is important for the problem of this thesis. A classification of routing systems, a static routing technique, effective insect-inspired routing algorithms and traffic models for simulation constitute the subsections of this chapter.

2.1 Taxonomy of routing systems

Vehicle Route Guidance systems implement strategies that discover paths used to direct vehicles from a chosen origin to chosen destination in a roadway network. Route guidance systems are projected to enable a driver to take the route that matches most diligently his requirements. Usually, this means the route that is shortest, fastest or least congested, or some combination of these should be selected. While systems vary in many aspects, all want to produce an optimal result from the user' and roadway network point of view.

Since we want to investigate the state of the art in traffic routing algorithms, we first need to classify and make a distinction between the key types of route guidance systems. Route guidance systems have been classified in different ways in literature. In particular, Schmitt and Jula [56] describe the characteristics of static and dynamic, deterministic and stochastic, reactive and predictive, and centralised and decentralised route guidance systems. In the following lines we will detail more the distinctions made by these two authors or by other authors. In addition, Kassabalidis et al. [44], classify routing algorithms as minimal and non-minimal referring mainly to telecommunication networks. We will also describe this.

Static versus Dynamic Systems

One important difference is whether a system uses a *static* or a *dynamic* approach for routing. The term dynamic is usually used when referring to a network or a routing algorithm/system. Di Caro [32] uses it also in relationship with the construction of routing tables. This comes from Dorigo et al. [25] characterisation of routing as the "network-wide distributed activity of building and using routing tables to direct data traffic".

The following lines detail in a few words the semnification of *dynamic*. The majority of routing algorithms/systems imply a network of connected nodes. As far as networks are concerned, the dynamic factor can be the either the fluctuating cost between nodes or the variation of the topology associated with node or link failures. The later is significant for telecommunication networks. Whether an algorithm is adaptive enough to cover these oscillations in a network makes it dynamic. Otherwise the routing policy is static. What is more, Bonabeau et al. [11] mention the existence of "quasi static" routing, where the path is modified only in response to exceptional events and/or on a long time scale. He describes dynamic systems as systems where entities in the network have an active participation in measuring traffic and network states.

Regardless of the changes in traffic characteristics during a period of time, a *static routing algorithm* maintains the same path for a user from source to the selected destination, [56]. Static networks presuppose that network conditions are time-invariant and the path is determined only on the basis of source and destination. Because of this, real-time changes in this system are not considered. Often this is called the offline routing approach. Static routing is based on simple table mappings established before routing begins. Such routing algorithms can react to changes only if the tables are altered according to the traffic patterns that are present in the network. Opposite in the vision of [56], the path-planning that incorporates real-time traffic information and reacts to fluctuating conditions of the traffic flow is called dynamic path-planning. Updates are usually implemented as the user moves towards his destination point. Nonetheless, this type of route guidance necessitates determining the current position of the user.

It is argued that dynamic routing is more desirable since is more robust than static routing, which can not be updated if congestion builds up. Besides, a dynamic variant where only the immediate link to be traversed is computed can be computationally efficient. While static route guidance has fewer requirements and demands less computation power, the system will be vulnerable to congestion. Dynamic routing will be more robust but more computational demanding.

Deterministic versus Stochastic Systems

Routing systems can be classified depending on the way the travelling cost is treated, as deterministic or stochastic [56]. The travelling cost is typically the travel time. *Deterministic routing* systems presume deterministic parameters for links and disregard the random nature of traffic flow. Even though the parameters can be updated periodically, the deterministic computation of the path results in the total distance that will be traversed by a vehicle. As a result, the total travel time will be computed using average vehicle speed without considering any stochastic variation of traffic.

In *stochastic routing* the random nature of traffic condition is considered and the road is regarded as a stationary or non-stationary variable. Stationary stochastic routing assumes the mathematical expectations (mean, variance) of conditions considered in the algorithm are time invariant, while the non stationary routing takes into account the time variance of traffic conditions. For non-stationary routing, mean travel time and variance are a function of time.

Again, routing based on more data, especially historical data has the disadvantage of being computationally more expensive, but allows stochastic systems to react better to the stochastic nature of congestion. This is to say that deterministic routing algorithms are mediocre compared to stochastic ones.

Reactive versus Predictive Systems

Reactive routing, known also as feedback routing is based on current conditions of the travel network. While such an algorithm reacts to real-time traffic information there is no insight into the future. Even though these kinds of systems are less complex than

predictive systems, they are more susceptible to congestion and incidents. On the other hand, *predictive routing* or *proactive routing* is based on anticipated conditions derived from a predictive model for a future time horizon that functions based on available historical data. This classification is based on [56]. But reactive routing algorithms are also named adaptive routing algorithms because they can adapt the routing policy to time and spatially varying traffic conditions [23]. Oscillations in the selected paths and large fluctuations of the measured performance are among their characteristics.

Despite that reactive and predictive route guidance systems are different, each can offer a real advantage for path planning systems. It depends, however, on what is most desirable: low complexity or robustness.

Centralised versus Decentralised System Architectures

Within a *centralised system* architecture, a main controller is responsible for monitoring the dynamics of the whole network in the process of planning a route (updating the routing tables, making routing decisions) and broadcasting the routes to all participants for which they are destined. While in a *decentralised* or *distributed* routing architecture, the computation of the route takes place for each vehicles independently and routing tables are also updated for each vehicle [23, 56]. In both centralised and decentralised routing systems, information about current and predicted traffic may be provided.

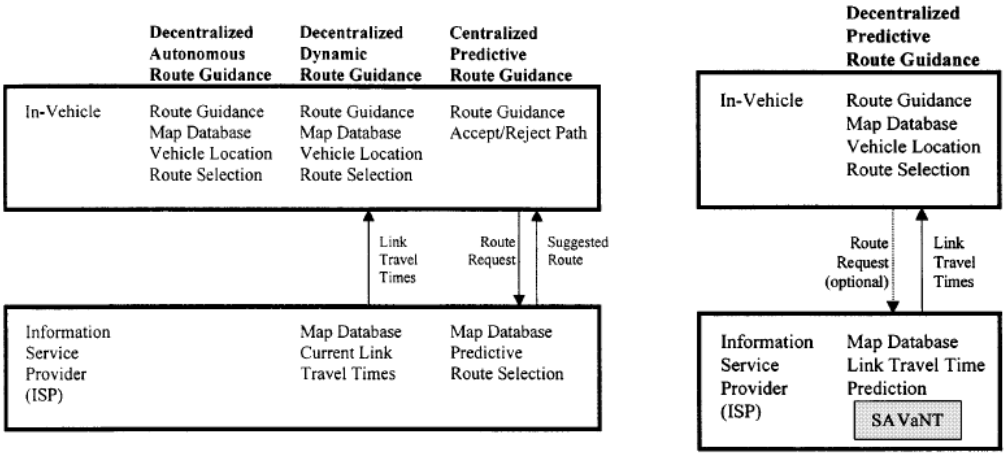


Figure 2.1: Example of routing guidance architectures according to [74].

Centralised routing can provide reliable routing for the complete network, according to [56]. Greater control over routing decisions and better visibility of the results is promised by a centralised routing system. However, in case of central controller's failure it is not possible to rely on central routing to have a rapid restoration of the paths. The distributed routing system, on the other hand, will not have this problem because is more robust. Distributed architecture of the routing system can encourage path diversification and, what is more, is not sensible to failure. Nevertheless, it is known that centralised systems suffer from scalability issues, that is the ability to adapt to a different network size. Moreover, delays necessary to gather information about the whole network status make them unreasonable in practice.

Wunderlich et al. present in the introduction of their paper [74] several examples of route guidance architectures (see Figure 2.1). Two interesting ideas are brought to light there. First idea argues that centralized routing is better than decentralized routing simply because allocating alternative routes over a decentralized route guidance architecture is governed by instability. In an architecture where the route-selection function

is located in-vehicle, route assignment (that might be desirable in some situations) is precluded. Allocations of routes is more stable in a centralised architecture because the central route computation unit can control more precisely the number of vehicles routed on a specific route. What is more, centralised routing is associated with more precise control over individual vehicle routing. The second idea argues that the evolutionary step which move routing systems from centralized to decentralized may not be necessary. A variant called Predictive Decentralized Route Guidance that includes link travel time prediction may be more rational to use and more appealing for users. This variant is also the one discussed in detail in [74]. An example of of a predictive approach consistent with a decentralized architecture is the Simulation of Anticipatory Vehicle Network Traffic (SAVaNT) developed at the University of Michigan.

Another interesting decentralized system is proposed by Xu [75]. Even though in some decentralized systems traffic sensors and vehicles are separate entities, in [75] the situation is different. Every participating vehicle in traffic is supposed to be equipped with an on-board integrated device responsible for sensing, collecting, analysing and disseminating traffic information. The functionality that in a centralized routing system is implemented by a traffic management center is now handled by each individual vehicle. Based on the information sensed by itself or received from neighbouring vehicles and based on the position and velocity a personalised route is computed. Said differently, this system consists of a large number of highly mobile sensors, namely vehicles. Transforming vehicles into sensors doesn't require additional infrastructure, but in the hypothetical situation that there are no such "witnesses ", traffic information is not available. Moreover, because traffic information needs to be processed and analysed by the vehicles themselves a powerful computation unit and a wireless inter-vehicle interface should be accessible. Redundancy of the traffic information, that comes from vehicles is considered critical for reliability.

Minimal versus Non-Minimal

Minimal routing and *non-minimal routing* are mentioned in [23, 44] for telecommunication networks but can be equally applied for routing cars. The first type of routing assumes that packets follow only minimal cost paths, while non-minimal routing allows more flexibility in choosing the path by utilising other heuristics. Minimal routing is also further subdivided in: *optimal routing* and *shortest-path routing* [21, 44].

Optimal versus Shortest path routing

The purpose of the latter is to minimise the cost of the path between two nodes in a disjoint calculation for all network and with no prior knowledge, while in the former: optimal routing , the objective is to minimise a function of all the link flows.

Reports from literature and notions inherited from game theory clearly state the difference in performance between the optimal and the shortest path routing. Since shortest path is widely used in routing algorithms, it is useful to understand how much the routing that is delivered deviates from optimality. Optimal routing has a global view while it assumes apriori knowledge about traffic patterns and is extremely important from the theoretical point of view because it provides a solution that is globally optimal, according to Di Caro [21, p.182]. Moreover, optimal routing models are called *flow models*. They have also been titled *multi-commodity flow problem*, "where the commodities are the traffic flows between the sources and the destinations, and the cost to be optimized is a function of the flows, subject to the constraints of flow conservation at each node and positive flow on every link" [21]. Flow conservation can be precisely defined only if the arrival rate of the cars on the links is available. The strategy inferred in solving the optimal routing make use of multiple paths for the same traffic flow and implies splitting the source-destination traffic at strategic points and shifting traffic among the

alternative routes resulted. The essential part of this solution is a spatio-temporal link capacity allocation among the flows.

Optimal routing is static and requires knowledge about all traffic characteristics. Shortest path algorithms are more flexible. One of their advantage is that the link costs can be computed statically or dynamically following some link states. Shortest path algorithms can be further subdivided in *distance-vector* (e.g. Bellman-Ford algorithm) and *link-state* (e.g. Dijkstra's algorithm), taking into account the content of each routing table.

However, the terminology is not supported in the same way by researchers. Moreover, this classification it not the only description of a routing systems.

2.2 Dijkstra's algorithm

If we ignore the fact that a real traffic road network represented through a graph can have *variable edge weights*, we are left with the **static shortest path routing problem**. This can be easily solved through a **static routing algorithm**. A disadvantage appears, however, when a change is made in the graph. Then the shortest path requires a complete recalculation. What is however disappointing, is that static routing is commonly used in navigational systems to the detriment of drives that are not rerouted when congestion occurs. For large dynamic networks static routing may be **time consuming**, because it recomputes everything once the data is updated.

Several contributions to the static shortest path finding problem were made by Bellman-Ford, Dijkstra and A^* algorithms. The A^* algorithm is a generalization of Dijkstra's algorithm that reduces the size of the sub graph that must be explored, if additional information is available that provides a lower bound on the "distance" to the target. The emphasis in this section lays, however, on details of Dijkstra's algorithm [18].

Dijkstra's algorithm solves the single-source shortest path problem for a weighted, directed graph with non negative edge weights, producing a shortest path tree. The search pattern can be visualised as an expanding circle around the source node that expands until it reaches destination. The assumption made for this algorithm is that the set of adjacent vertices is available for every node. In the algorithm shown in Algorithm 1, V is the graph edges (see page xi), w a function that determines the weight between two adjacent vertices and s is the source.

Algorithm 1 Pseudo-code of Dijkstra's algorithm.

```
Dijkstra(V, w, s)
  {V -the set of nodes in a graph}
  {w -weight function of an edge}
  {s -source intersection}
  Initialize(V, s)
  S ← ∅                                  {Set of all visited cities}
  Q ← CreateEmptyList()                  {Set of all unvisited cities}
  AddToList(Q, V)
  while Q ≠ ∅ do
        u ← ExtractMin(Q)
        AddToSet(S, u)                   {adds u to set S}
                                         {for all neighbours of u}
    for all v ∈ N_u do
              Relax(u, v, w)             {where v has not yet been removed from Q}
    end for
  end while
```

Algorithm 2 Pseudo-code of procedure *Initialize*, Dijkstra's algorithm.

```
Initialize(V, s)
  for all v ∈ V do
        d[v] ← ∞              {Unknown distance from source to node v}
        previous[v] ← ∞       {Previous node in optimal path from source}
  end for
  d[s] = 0                    {Distance from source to source}
```

Algorithm 3 Pseudo-code of procedure *Relax*, Dijkstra's algorithm.

```
Relax(u, v, w)
  {Determine if current route to v via u is shorter}
  alt ← d[u] + w(u, v)
  if alt < d[v] then
      d[v] ← alt              {Relax the weight of node v}
      previous[v] ← u         {The previous node on the shortest path to v is u}
  end if
```

Dijkstra's algorithm maintains a set of *visited vertices* S for which the final shortest-path distance from source, $d[v],\ v \in S$, is going to be determined. Intially, S is empty but contains all nodes once the algorithm is finished. The algorithm repeatedly selects one of the unvisited vertices u in Q with the minimum shortest-path estimate. Q is a priority queue. This vertex is added to S and all edges leaving u are relaxed. Relaxing (see Algorithm 3) is testing whether the shortest-path estimate of a vertex can be lowered by determining if the current edge is the best edge to reach this vertex. If the shortest path estimate $d[v]$ can be lowered, the best edge to reach the next node is set then to u. The latter implies setting the $previous[v]$ to u.

As was argued earlier, Dijkstra's algorithm, like any other static routing procedure is not indicated for dynamic and adaptive traffic routing. Computation time is high when the network is large because at the slightest change in costs the shortest path has to be recalculated. Dijkstra's algorithm offers a pre-trip routing advice. On the negative side, Dijkstra's algorithm is a classical example of algorithm applied in a **centralised routing** architecture.

2.3 A swarm intelligence approach to routing

Swarm Intelligence, an active area of research in recent years, has as a major emphasis the "design of adaptive, decentralised, flexible and robust artificial system, capable of solving problems through solutions inspired by the behaviour of social insects" [10]. Because our motivation and goals for this thesis are quite similar, swarm intelligence is an area of great interest.

The aim of this section is to acknowledge the role of biologicaly inspiration in the design of several algorithms relating to the shortest path problem. In the following pages, we will show the rationale, the advantages and the disadvantages that lie behind the mentioned methods.

2.3.1 The class of ant "inspired" algorithms

First we will take a look at field of *ant algorithms*, where primarily mathematical models have been developed based on the observation of real ant behaviour [20]. These mathematical models are further used in solution optimisation. The study of ants has been very appealing to scientists for a long time due to the characteristics their colonies exhibit:

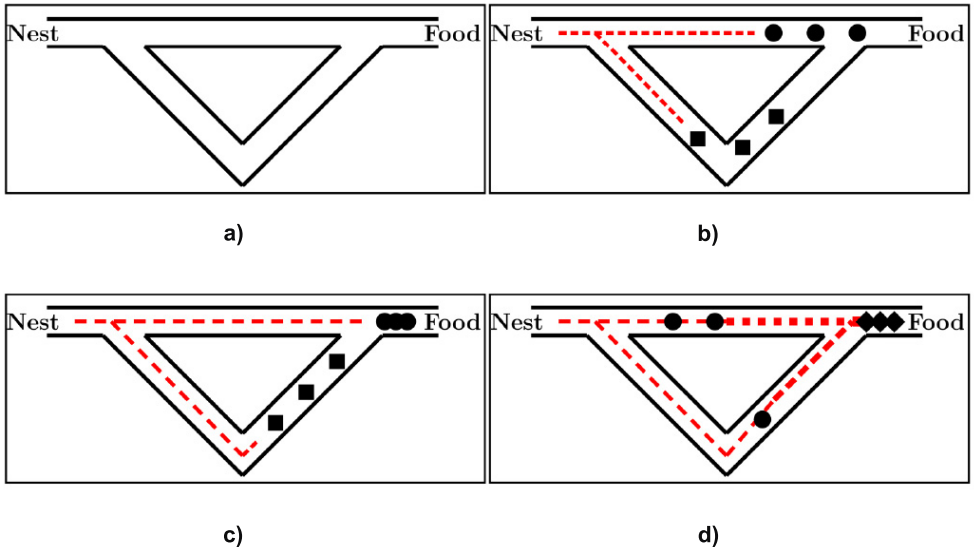


Figure 2.2: Shortest path finding capability of ant colonies.

Shortest path finding capability of ant colonies. Pheromone trails are depicted as dashed lines and thickness indicates strength, [6]. (a) All ants are in the nest. There is no pheromone in the environment. (b) The foraging starts. In probability, 50% of the ants take the short path (symbolised by circles), and 50 % take the long path to the food source (symbolised by rhombus). (c) The ants that have taken the shortest path have arrived earlier at the food source. Therefore, when returning the probability to take again the short path is higher. (d) The pheromone trail on the shorter path receives, in probability, as stronger reinforcement, and the probability to take this path grows. Finally due to the evaporation of the pheromone on the long path the whole colony will in probability use the short path.

fully distributed control, environment-mediated communication, cooperative and collective strategies, and self-organisation. These features were the origin of new algorithms and multi-agent systems. We hope that the review of this technique will bring more light in developing a routing algorithm according to what was stated in Section 1.2.

ACO is a relatively new approach developed in the early nineties in the field of swarm intelligence as a general purpose optimisation technique. Taking inspiration from the foraging behaviour of Argentine ants *Linepithema humile*, ACO is a population based meta-heuristic that can find approximate solutions to a large amount of problems. A meta-heuristic is an algorithmic framework used to define heuristic techniques that can be applied to an entire range of problems, with few adjustments. Heuristics are designed to gain computational performance or conceptual simplicity, potentially at the cost of accuracy or precision. Heuristics typically use problem specific information to cultivate solutions.

All ant colony optimisation algorithms share the same representative idea with biological ants. We are particularly interested in ants'capability of finding shortest paths and in the concept of marking paths with pheromones to solve this kind of problem. Ants are social insects that live in colonies and their behaviour is primarily governed by the goal of colony survival. In the real world ants wander arbitrarily until they find food sources and then return to their colony. In their struggle to find paths between nest and food source, ants deposit a chemical substance called *pheromone*. These insects don't travel at random and can smell previously deposited pheromone. When moving, ants chose the paths with higher pheromone intensity. At the same time, pheromone helps ants to return to their nest when coming from a food source. The intensity of the pheromone is a result

of repeated and simultaneous experiences of individual ants. However, pheromone used by ants slowly evaporates over time. Moreover, the information deposited in pheromone is available for the ants only locally. A short path is marched over faster than a longer path and more pheromone is put down on the shorter path. The previously described process is a kind of distributed optimisation mechanism in which each ant gives a small contribution [25]. To show how this can happen let us consider the case idealised in Figure 2.2.

2.3.1.1 Features of ant algorithms

Characteristics of real ants transferred to artificial ants

The idea behind the ant colony framework is to simulate in an iterative fashion the behaviour of a number of artificial ants moving through a graph that encodes the given problem. Nonetheless, the artificial ants preserve in a great measure the characteristics and behaviour of a real ant colony. Therefore, details over the features that create the starting point of ant algorithms will be further offered.

1. *Stigmergy and pheromone trail.* Stigmergy is a valuable concept used to explain the self-organising capabilities of ants as well as the capabilities of other social insects. Self-organisation requires interaction between insects. Such interactions are usually classified as direct and indirect. The research community defines stigmergy as an indirect and non-symbolic form of communication mediated by environment [24]. Two individuals interact indirectly when one of them modifies the environment and the other responds to the modifications of the environment at a later time [10]. For this, real ants deposit on the world's state it has visited some quantity of pheromone. Stigmeric information can be accessed by agents only locally. In addition, stigmergy does not explain in detail how individuals coordinate their activity but offers a more general mechanism for their behaviour at the colony level. One of the features that stigmergy brings to the field of optimisation is the incremental construction of a solution: a new solution is built from previous solutions. Additionally, stigmergy is valued for the flexibility associated with it as [10] point out. In this respect, artificial agents inspired by social insects are able to respond to perturbations even though they were not programmed to deal with certain changes in the environment. Stigmergy, the characteristic of real ants, can be extended to artificial ants according to [25] by:

   (a) Associating to problem states appropriate state variables. Artificial ants will modify some numerical information stored in state variables. Analogous, the pheromone released by an insect corresponds to a modification of the environmental states visited during solution construction.

   (b) And by giving local access to these variables' values. This corresponds to the local nature of pheromone released by ants.

2. *Autocatalytic behaviour.* While walking repeatedly from the nest to the food source and viceversa, real ants deposit substances called pheromone on the ground, forming a pheromone trail. Initially real ants move randomly. But ants that leave the nest at a later time guide their search by sniffing pheromone left on the ground by other individuals. In this way, the colony possesses a form of autocatalytic behaviour which promotes exploitation of positive feedback [30]. For artificial ants this is translated into: if more ants follow a certain trail, then the probability with which a new ant chooses the same path is higher. Said in a different way, pheromone trail has a bigger strength on the edges appearing frequently in good solutions [47]. Autocatalysis is an important mechanism for population based optimisation

algorithms. In evolutionary computation algorithms is implemented by selection and reproduction mechanism [38].

3. *Dynamic collective memory.* The experience gained in path determination by each single ant is encoded in the pheromone trails distributed on the ground. In turn, this pheromone influences every decision an ant takes. The pheromone trail itself acts as a collective memory, a shared "repository of the most recent foraging experiences" as Di Caro ([21]) calls it, which is permanently adapted by the travelling ants. Ants use a measure of goodness to mark each move they make in space.

4. *Distributed control.* The pheromone trails, once are built, assist the ant colony in selecting and following the path with the shortest time from nest to food source. In this sense, we can observe a form of distributed control based on indirect communication among agents. Even though the ants can be considered autonomous, the overall control is distributed [21].

5. *Implicit path evaluation.* Reinforcement on the shorter paths is a form of implicit path evaluation, because shorter paths are completed faster than longer paths and they receive reinforcement of the pheromone trail much more quickly than longer paths [21]. Even though there is an equal number of ants choosing either the shortest or the longer path, as in Figure 2.2, the shortest path is more attractive and has a higher pheromone strength.

6. *Pheromone biased stochastic decision.* Because pheromone is a local signal that encloses the previously accumulated knowledge of the shortest path, ants use a local stochastic policy to determine the next step whenever they reach an intersection of paths. Artificial ants make use of local information and there is no information about future states. This is a local policy in both in space and in time [25].

7. *Iterate concurrent computation.* Each ant contributes to the general behaviour of the colony through the small steps performed form source to nest. A single ant is capable of finding a path between nest and food, but it is only the simultaneous presence of other ants that converges to the optimal path [32]. Ants act concurrently and independently.

Distinct features of artificial ants

Ant algorithms are an abstraction of some behavioural patterns of real ant algorithms related to shortest path searching. Both real and artificial ant colonies are composed of individuals that cooperate for a certain goal, either to find food or in case of artificial ants to find a good solution to a problem. Nonetheless, artificial ants also possess several characteristics that do not have a natural matching part. These features permit them to achieve good solutions for the tackled problems.

1. *Discrete world.* Artificial ants live in a discrete world and their moves are transitions from one discrete state to another one [25].

2. *Synchronised simulation system.* Despite the fact that real ants move in an asynchronous way, the artificial ants are synchronised. At each iteration, the artificial ant moves from source to destination and then backwards [6].

3. *Pheromone's update mechanism.* Unlike real ants, artificial ants use explicit memory to store information about the followed path. The memory is the internal state of the ant, containing all information about past actions [25]. Next, this memory is used to build feasible solutions, to evaluate generated solutions, to retrace the path followed and to update the pheromone on this path. Some artificial ants only

deposit pheromone after generating a complete solution. But this kind of timing of ants is problem dependent. Once the solution has been constructed the pheromone trails are updated by retracing the travelled path by means of a process called online delayed pheromone trail update [6, 17, 32]. The pheromone deposited is a function of the quality of the solution found by every ant. Other, make use of online step-by-step pheromone trail update and update the pheromone trail each time they traverse an edge that is part of the problem representation [17].

4. *Pheromone's evaporation mechanism.* It has been proven in the experiments with foraging ants [20] that evaporation of pheromone is quite slow in a natural ant colony. To avoid the algorithm getting stuck in local optimum, pheromone evaporation is considered an important mechanism for artificial ant colony. Evaporation prevents premature convergence and favours more exploration in the solution space. If pheromone decay is slow, it is very probable that ants will get trapped in sub-optimal solutions, where not very best alternatives take over because the previous choices still influence in a great measure the current choices. On the other hand, if pheromone decreases fast, ants will take advantage of the accumulated knowledge for a shorter time. In general, pheromone evaporation is quite an important issue when multiple paths are available or when characteristics of the environment change dynamically. In this case is imperative to set the appropriate tradeoff between the exploitation of a path that seem particularly good and exploration of other alternatives, [21]. We can also say there is a correlation between the level of stochasticity present in ants' decision policy and the balance exploration/exploitation [25]. Together with pheromone deposit, pheromone evaporation has the purpose of focusing the search process of ants towards promising regions of the solution space. This is probably the most researched part of ant algorithms and many variations of the updating and evaporation mechanism have been proposed. We will detail some of the rules in the sections to come.

5. *Extra capabilities.* Besides the pheromone trails, artificial ants take advantage of additional problem specific heuristic information. Additionally, ant algorithms can be augmented with extra capabilities such as local optimisation and candidate list which contain the most promising neighbours to visit. Ant algorithms may also be augmented with the capability of observing the quality of all solutions generated and releasing additional pheromone amount for certain solutions.

2.3.1.2 Application of ACO to static problems

Algorithms that take inspiration from real ants behaviour in finding shortest paths have been successfully applied to several static optimisation problems. Further on, we summarize several successful algorithms that belong in this category and have been used for routing and other optimisation problems. A non-exhaustive list of successful ACO algorithms can be found in Table 2.1.

Ant Colony optimisation (ACO) is one of the most popular techniques for approximate optimization and state-of-the-art for many problems. ACO can be used for some kind of routing like TSP, VRP and routing in telecommunications networks. In ACO metaheuristic a colony of ants with the previously mentioned characteristics collaborate in finding good solutions to optimisation problems. ACO covers most of the existing ant algorithms variants for optimisation problems. ACO can solve both static and dynamic problems. One example of a static problem for which ant colony optimisation has shown great results is TSP, where no characteristics of the problem change over time. Nevertheless, problems related to telecommunications and transportation networks are distributed and non stationary. Here it is difficult to model the variability of traffic and/or network topology over time. In this case, any algorithm needs to be capable of adaptation to

Table 2.1: A selection of successful ACO algorithms for static CO problems.

Algorithms are listed by order of appearance. This table is adapted from [32, 21].

ACO Algorithm	**Main references**	**Initial Problem**
Ant System (AS)	[29, 30, 33, 31]	TSP
Elitist AS (EAS)	[29, 30, 33, 31]	TSP
Ant-Q	[39, 26]	TSP
Ant Colony System	[26, 27, 28]	TSP
Max-Min AS	[61]	TSP
	[60]	TSP,QAP,FSP
	[62]	TSP,QAP
Rank-based AS	[12]	TSP
(AS_{rank})	[13]	VRP
ANTS	[52]	QAP
BWAS	[16, 15]	TSP,QAP
Hyper-cube AS	[8, 7]	-

Abbreviations for types of problems: Problems are abbreviated by a combination of first letters of their name. TSP: Travelling Salesman Problem, QAP: Quadratic Assignment Problem, FSP: Flow Shop Problem, VRP: Vehicle Routing Problem.

fluctuating conditions. But ACO can be applied due to ants' concurrent way of functioning and to their adaptive nature. ACO is particularly useful for distributed stochastic problems.

ACO is a stochastic constructive procedure that incrementally builds a solution, [32]. By means of pheromone update and pheromone evaporation, ACO concentrates the search in regions where high quality solutions are found.

Applying ACO to a CO problem is quite simple. This framework requires a mapping of the problem which permits incremental construction of the solution, a neighbourhood structure and stochastic transition rule.

ACO uses a pheromone model to generate solutions to the problem at hand by assembling candidate solutions from solution components. The set of all possible solution components denoted by $C = \{c_1, c_2, \ldots, c_n\}$ is consequent to the CO problem under consideration. In ACO , artificial ants construct a solution by building a path in a construction graph $G = (C, L)$, where L are called connections because they connect the elements in C. Each solution component has associated a pheromone value τ_i and heuristic information η_i which reflects the desirability of a component. A larger pheromone value in combination with a large heuristic value corresponds to a higher probability of making a certain choice in component selection. Comprehensive details about the construction of solutions will be provided at a later time.

In the case of AS applied to TSP, a solution component can be considered either an edge or a vertex. Artificial ants build partial solutions by traversing the fully connected construction graph $G(V, E)$, where V is a set of vertices and E is the set of edges obtained from the set of solution components C. After bring together candidate solutions, this candidate are used to modify the pheromone values and bias the search towards high quality solutions. However, an ACO assumes that good solutions are created from good solutions components [6]. Whether this is true is debatable.

The ACO framework consists of the scheduling over the following procedures: ant based solution construction, pheromone update and daemon actions. Sequential scheduling for the three procedures can be exploited in the case of non distributed problems, while for distributed problems such as routing in telecommunication networks some parallelism is necessary. In non-distributed problems, knowledge is accessible at any instant. The generic running of the earlier mentioned processes is represented in Algorithm 4.

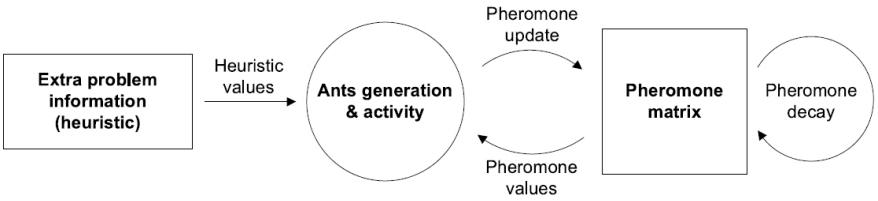


Figure 2.3: Process organisation of the Ant Colony Optimization Metaheuristic.

Daemon's actions are not listed because are generally external to the cyclic process, [2]. The presence of an arrow indicates the existence of an interaction between blocks, while the arrow's direction represents the direction of the main information flows.

Algorithm 4 Structure of ACO metaheuristic in pseudo-code.

The procedure DaemonAction is optional and refers to centralised actions executed by a daemon at a global level. Adapted from [6, 32].

```
Procedure ACOMetaheuristic
Set parameters, initialise pheromone trails
while termination conditions not met do
  ScheduleActivities
  ConstructAntSolutions
  UpdatePheromones
  DaemonActions{optional}
end while
```

Still, most ACO algorithms employ a more specific algorithmic scheme than the general ACO metaheuristic framework given in Algorithm.

ACO is an iterative algorithm controlled by a while loop. In each iteration the algorithmic components are scheduled and synchronised depending on the preference of the designer. At each iteration a number of solutions are constructed by ants, which are then used to update pheromone trails and additionally daemon actions may be performed at a global level.

ACO algorithms have emerged from several pheromone update schemes. As a result of this many aspects of the ACO are implementation-dependent. One first example in this respect is pheromone update. This can be either online step-by-step or offline delayed. The variants of updating are, however, mutually exclusive and cannot miss both at the same time. Pheromone updating includes pheromone evaporation and pheromone accumulation, but pheromone rules are variations of a general rule.

Daemons are also implementation-dependent and differences on this theme exist, between the best known ACO algorithms. A daemon's task is to collect additional information and deposit pheromone that will bias ant's search from a non-local perspective. A daemon may also apply local search methods to already constructed solution [6].

A visual representation of the process that takes place inside an ACO framework is represented in Figure 2.3.

2.3.1.3 Application of ACO to dynamic problems

As many other heuristics, ACO algorithms have proven their value in static environments, but less is known about them in dynamic environments. As it is said in [36], the problem in this kind of environment is not finding desirable solutions, but tracking them in the solution space. Dynamic or adaptive routing has been a topic of significant interest in the past years. The efforts were concentrated on telecommunication problems as well as vehicular traffic routing. Further, we discuss several important algorithms for the field of dynamic routing.

Ant Based Control (ABC) was the first successful swarm based routing algorithm designed for telephone networks [57, 58]. The central task for this type of networks is to allocate calls over multiple switches so that the network can support a maximum number of possible calls during peak hours. Equally important is that no congestion will appear and no calls may fail. Even though results can not be judged on a large scale, ABC is important because has stimulated the curiosity of ACO researchers for dynamic problems.

Moreover, ABC is designed for special kinds of networks were certain assumptions are made. Many implementation details are bound to these suppositions. In [58] the network in which the routing takes place is modelled by a graph. Additionally, each node has attached a total capacity C_i, a spare capacity S_i and a routing table R_i. Links (i, j) have attached a vector of pheromone values $\tau^i_{j,d}$, to each destination d, which represent the desirability of going from i to j. One of the first assumptions of ABC refers to these connection links that can potentially carry an infinite number of calls. The second assumption made when the network model was defined is that transmission nodes/switches have limited connectivity. Interesting enough, due to these considerations bottlenecks occur in nodes and congestion is defined as the number of connections available at each switch. This assumptions can not be considered after all for road traffic networks.

Ants can be launched from any node at any time with a randomly selected destination. Their role is to update R_i tables. In their wandering, ants use the value of the routing tables corresponding to the current node and the corresponding destination as a probability to find the next point in the path. A route for a call from node s to node d is built at set-up time by choosing sequentially and deterministically the neighbour node with the highest probability value, until d is reached. Ants choose the neighbour to move to following a uniform random scheme over current neighbours.

Reminiscent of Ant System of Dorigo et al. [30], Dorigo [33], ants update pheromone through an *on-line step-by-step procedure* only on the visited links. There is only one class of ants and the update will be used by ants moving in the opposite direction of the updating ant. Here, another assumption is made, namely that links are bidirectional. However, this does not hold for transportation networks.

Some of ABC's particular characteristics are that ABC does not use local traffic models, nor additional heuristics and neither a memory to improve decisions policies and to avoid cycles. The ants of ABC do not also use the information found in sub-paths. Moreover, an initialisation phase is required for the values of the routing tables.

Schoonderwoerd et al. used techniques like: ageing, delaying and noise to avoid stagnation in the probability values of the routing tables. The rate at which the ants are transmitted from congested nodes is reduced and due to ageing mechanism they deposit less pheromone on the nodes they subsequently visit. For the other links that do not belong to the current route, a normalisation of the pheromone values is operated. Ants also grow older after each node hop. Through this mechanism, pheromone deposited by ants is inverse proportional to the length and degree of congestion of the path. Delayed ants have a bigger age so they don't influence very much the routing tables. By limiting the ants launched from every source to one, the authors of ABC hope to eliminate circular routes. We refer the interested reader to search for more details about this reinforcement parameter in [58].

In [11] a significant enhancement of pheromone updating was proposed. Bonabeau et al. extended ABC with the idea of "smart" agents. The ants update the probability values at an intermediate routes in addition to their main route. This can be applied both to one-way ants and to ants that perform round trips. Compared to the ants of the original ABC smart ants exhibit a more complex behaviour with a reduced number of agents in a way that reminds of the dynamic programming principle of optimality. The

number of ants is reduced because they make more pheromone updates.

Other improvements are reflected in [63] and [42]. Subramanian et al. [63] design two types of ants: regular and uniform, that update pheromone differently. Regular ants update the pheromone values based on the accumulated cost of travelling to a node. On the other hand, uniform ant select randomly the next intersection and modify the pheromone values in the opposite direction of their travel, but also based on their costs. One assumption of the algorithm implies that all the neighbours costs are known. Heusse et al. [42] have proposed a cooperative asymmetric routing model for packet-switching networks. Together with introducing asymmetric path costs, the authors maintain at each node a table that records the cost for every possible destination via any possible neighbour. Still the algorithm's performance degrades under low traffic.

Hierarchical ABC (H-ABC) is an extension of ABC proposed by Tatomir and Rothkrantz, first for telecommunications networks [65], and then for routing in vehicle traffic networks [64, 49, 66]. But our interest is focusing mainly on how vehicle traffic routing is performed in this approach. H-ABC, while being an ant colony inspired algorithm, is also a hierarchical routing method. Therefore, we postpone the discussion about it till we reach Subsection 2.3.3.

AntNet A very successful application of ACO to distributed and traffic-adaptive multipath routing problems is AntNet algorithm, first proposed in [22, 23]. AntNet was designed to help solving the routing problem for packet-switched networks. Moreover, AntNet is a state-of-the-art method that has outperformed a set of algorithms on numerous benchmarks under different traffic patterns [21, 23].

Inspired by the foraging behaviour of ant colonies, AntNet continues the work of [58, 57]. While ABC uses *online step-by-step pheromone update* strategy, AntNet uses *online delayed pheromone update*. AntNet is one example of swarm intelligence algorithm that incorporates round-trip agents. Informally, AntNet's steps can be summarised as follows.

The whole algorithm's behaviour is based on the complex interactions of two types of mobile exploration agents: the Forward and the Backward ant, that realise a sampling and an update of information about the paths connecting sources to destinations. Forward ants act as investigators, while backward ants modify routing tables, according to the description of [44]. Routing is determined through complex indirect communication between ants, called stigmergy. The exchange of information is realised through the update of data structures called routing tables. At regular time intervals, forward ants F_{sd} are launched concurrently and asynchronously from each network node s towards a destination node d, using the same queues as data packets. In this way the time elapsed while moving from source to destination can be considered a measure for the quality of the path. One of the specific tasks of a forward ant is to find the minimum delay path connecting s to d by moving from node to node based on a stochastic decision policy. However, the forward ant does not modify any routing table, since this is the task of the backward ants. The forward ant is equipped with a memory stack, where the visited node is pushed together with the entrance time. The trip time values are computed by taking the difference of two adjacent nodes pushed on the stack. If cyclic paths are detected, the cycle's nodes are removed from the ant's memory. But if the cycle is longer than the ant's life before entering the cycle, the ant is destroyed. In this way, only the information that comes from reliable ants is used to update tables. A trade-off between the local pheromone variable, the status of the links' queues and the information carried in ant's memory take part in selection of next hop node.

Once arrived at the destination node d, the forward ant transfers all of its knowledge to a backward ant B_{ds}. The backward ant goes back to the source node s using priority

queues faster than those of forward ants, in order to quickly update the routing tables of each node and the local models of the network. At every node, a parametric statistical model for the traffic distribution over the network retains the average trip time, the best trip time and the variance of the trip times for each destination. The amount of pheromone deposited by the backward ant is proportional to the goodness of the path built by the forward ant during a couple of iterations. If the travelling times on the ,sub-paths followed by an ant F_{sd} corresponding to the backward ant B_{sd} are good, updates are performed in the local model and in the routing table also for the sub-paths. In fact, sub-paths can be considered secondary evaluations of other routes discovered, at no cost, by ants. This mechanism is different from that adopted by Schoonderwoerd et al. for cost-symmetric networks ([57, 58]) in the respect that it uses additionally a backward ant. By following this approach, AntNet supports path completion and evaluation before carrying out an update.

When arriving back at the source node the backward ant is removed from the network. Data packets are then routed based on the information contained in the routing tables, also using a stochastic decision policy.

Baran and Sosa have proposed in [4] several **improvements over AntNet** which have increased its performance. The most significant are: intelligent initialisation of routing tables, a restriction on the number of ants and a mechanism to handle pheromone update after node failures.

Liang et al. [50] have studied what impact has the routing table size on the performance of AntNet. They have designed a new algorithm termed **AntNet-local** that reduces the number of entries in the routing table. In the original AntNet, an assumption is made and that implies the use of global routing table information. Specifically, at every node if there are n outgoing links and d nodes in the entire network, the routing table will have $n \times d$ entries, while in AntNet-local there will be only $n \times 2$ entries. However, experiments that compare AntNet-local [50] with AntNet-global show that the second one is much more desirable and of higher performance. This motivated the same authors to develop a Distributed Genetic Algorithm (DGA) [51] to eliminate the need for having an entry for each destination in the routing table. For the new algorithm Liang et al. have used the principles of genetic Adaptive Routing Algorithm (GARA). In the approach proposed in [51], ants perform a traversal of a set of nodes known as *chromosome*. Once that operation is completed, the agent becomes a backward agent that returns to the source and modifies only the routing tables belonging to the source. The source measures the fitness of the agent and generates a new population. Routing is performed based on the routing tables through the shortest path. There is, nevertheless, a degrade in performance as the penalty paid for reducing the routing table's size which makes DGA achievements situated between AntNet and AntNet-local. Please refer to [51] for more details.

As an overall remark, **ABC** and **AntNet** are quite similar and exhibit both adaptivity and robustness.

The dynamism of the routing problem is also treated by Eyckelhof and Snoek [36] but in a different manner. They handle changing travel time between cities with a **new Ant System** (AS) algorithm and several ways of adapting the pheromone matrix both locally and globally. What is distinctive for this AS is the strategy of smoothing pheromone values in the area containing a change. One of the main effects targeted by [36] is how exploration is performed by ants. In dynamic situations, solutions that are not good before a change in the environment might turn out to be good afterwards. Measures need to be taken to make ant's exploration aware of the changing conditions. Improvements suggested are: a lower bounder for the amount of pheromone (or reset) and a *shaking* technique for smoothing pheromone levels that insures nevertheless that relative ordering of best links is preserved. Other alternatives for applying shaking strategies in

the pheromone matrix were offered by Guntsch et al. in [41].

In [36], tests were conducted on the Traveling Salesman Problem (TSP) which has been extended with the incorporation of incremental simulated traffic jams. Traffic jams usually cause the associated travel time to increase. It has been concluded that AS perform good on dynamic problems and that local shake made a significant difference. The higher the frequency of the dynamics is, the more important is to preserve earlier information. Equally beneficial is a combination of exploitation and biased exploration.

2.3.2 The class of bee "inspired" routing

A second approach from the swarm intelligence field, that has demonstrated a similar or better performance compared to state-of-the-art ant algorithms are *bee hive inspired methods*.

2.3.2.1 Bee colony's characteristics

Bee colonies have similar foraging characteristics to ant colonies. However, there are several distinct features that need to be acknowledged.

1. *Selective reinforcement.* Honey bees evaluate each potential food site by performing a waggle dance in front of their fellow foragers. If the quality of the food source exceeds a certain threshold, the direction and the distance to that site is encoded by the specific "dance". So not every discovered site receives reinforcement.

2. *Blackboard communication model.* In comparison to ant algorithms, where the agents use the principle of stigmergy for communication, in a bee hive, a blackboard system is responsible for mediating the communicative effort.

3. *Short and long distance bees.* The majority of foragers is concentrated on exploring the food source that are found in the vicinity of their hive, while only a minority visits sites faraway from the hive.

2.3.2.2 Applications of bee colonies

BeeHive is another routing algorithm that borrows its main ideas from social insects, namely from bee hive communication. Wedde and Farooq have been the fist to develop multipath routing techniques for packet switching networks [71] based on the principles that govern bee's behaviour. BeeHive is a hybrid approach between flat and hierarchical routing algorithms.

In this algorithm, the network graph is partitioned into zones called *foraging regions* that result based on particularities of the topology. Each region has an elected *representative node*. If this node crashes is replaced by another one. Besides, the foraging region, every node has also its own *foraging zone*, which consists of all nodes situated at a maximum number of hops from this node. Through this partitioning, the adequate environment for implementing short and long distance bee agents is created. Short distance bee agents collect and disseminate routing information in the neighbourhood of the source node, while long distance bees have the freedom to gather the necessary routing information to all nodes of the network.

Non-representative nodes launch short-distance bee agents, by broadcasting *replicas* to the neighbouring nodes. Representative nodes launch long distance bee agents, which have a bigger lifetime limit, or they are allowed to go a bigger number of hops whenever exploring routes. The replicas do an exploration using priority queues and they deposit the trip time and the queueing delay for reaching its source node from the current node, over the incoming link. However, replicas are not sent to neighbours from where they

arrived. The process continues until either the lifetime of a bee has expired or the replica of a bee has arrived at a site.

As opposed to AntNet, the algorithm uses just forward agents and no statistical parameters are stored in the routing tables. An estimation model computes the trip time from a source to a given node. Moreover, an advantage is that the size of the routing tables is reduced as compared with AntNet. The need to retain $O(n*d)$ entries (where d is the number of possible destinations and n is the number of neighbours of node i) for the routing table of node i, as in AntNet, is eliminated by the introduction of three routing tables. Each node maintains information for reaching the nodes in its foraging zone via an *Intra Foraging Zone* table (IFZ) and for reaching the representative node belonging to different foraging regions via an *Inter Foraging Region* table (IFR). Additionally, there is a *Foraging Region Membership* table (FRM) that maps known destinations to corresponding foraging region. Analogous to AntNet, data packets are routed in a probabilistic manner according to the quality assigned to current's node neighbours. This helps maximising system's performance even though data packets will not always follow the "best" path. Nevertheless, the big disadvantage consists in the quantity of memory used to distinguish between different replicas.

Last, any further details about BeeHive can be found by the interested reader in [71].

BeeJamA proposes a layered approach algorithm inspired by honey bee communication for on-line routing of single cars and congestion avoidance before it occurs, [72]. This is the first attempt of this kind. The new algorithm is decentralised, multi-agent and highly adaptive. The authors wish to cope with individual travellers' need for the shortest route possible and to keep a low profile of congestion in the whole system. BeeJamA is an extension of BeeHive used in telecommunication networks, [71] and encloses a graph called *area layer* that matches the actual road network and a *net layer* formed of areas of the previous network as in Figure 2.4.

In general, a vehicle can go from source to destination in the same area or needs to reach a destination in a different area. First the vehicle's route to destination is selected on the net layer. For routing on the net layer, the network is partitioned in *foraging regions* and each node has its own *foraging zone* on the net layer. Three routing tables are present for this layer, that remind us of BeeHive: intra foraging zone table (IFZ_{net}), foraging region membership table (FRM_{net}) and inter foraging region table (IFR_{net}). The routing tables in the net layer follow the same structure as in BeeHive. Moreover, their updating is similar.

Second, after the vehicle's route to destination is selected on the net layer, the vehicle has to be routed on the area layer. In the routing process the responsibility falls on a Navigator that manages routing tables and checks the position of every vehicle in the area. All areas are connected by *border nodes*. But the routing needs some adjustments, as compared to standard BeeHive. Again, three types of tables will be used, however these are different from those belonging to the net layer. Namely, the routing tables are: IFR_{area}, IFZ_{area}, FRM_{area}. On the area layer, IFR_{area} is different from the IFR_{net}. In the area layer case, this table is equal for all the nodes in a specific area layer because for all the nodes in an area the foraging zone corresponds to the respective area.

What is different from BeeHive is the edge cost's estimation model that takes the form of a quality rating function. Starting from density-speed curves, empirically approximated, speed is computed for three separate density intervals. Then the cost, meaning the time to traverse an edge is easily determined from length of the edge and speed. A route is not selected if travel cost does not exceed a certain threshold.

On the whole, this algorithm routes cars from intersection to intersection "on a next-hop basis". The nodes belonging to the optimal route are not known in advance, but just before reaching the next vertex. Differently from BeeHive, the intersections do not

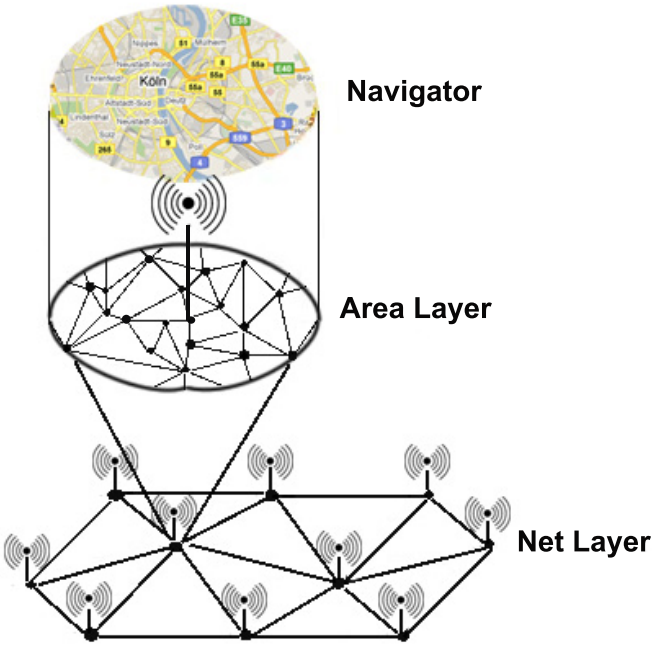


Figure 2.4: Layered routing model of BeeJamA according to [72].

store routing tables and they communicate with the incoming or crossing vehicles. In BeeJamA, vehicles are directed in a rough direction to their destination at the beginning of the routing process. After they have reached the neighbourhood of the destination the routing becomes more precise. Decisions are made probabilistically based on travel times across edges.

In [72] a comparison of BeeJamA with Dijkstra's algorithm, which benefits from traffic updates at every 10 minutes, has been realised. Several simple scenarios and a realistic one for the Ruhr area were tested with traffic generated by a stochastic and discrete traffic simulator based on a cellular automaton model. The full implementation and analysis details are to be found in [48].

2.3.3 Hierarchical nature-inspired routing

H-ABC was designed in [66], as mentioned earlier. The main motivation behind developing this algorithm was the scalability issues of ABC. Even though ABC has a good performance on small networks, the performance is not the same for big networks.

Tatomir and Rothkrantz have imposed a split of the vehicle traffic network into clusters of less complexity called *sectors*, which introduce a hierarchy of roads. The upper level of the hierarchy is occupied by *global sectors*. Besides, in the hierarchical network routing is performed by a Global routing system and a Sector routing system. In the detail level sectors, nodes at the border of the sector play a special role and are called routing nodes. For every sector, a virtual node abstraction is introduced that will route cars between different sectors. Moreover, every node of the physical network has a routing table associated with it. In addition to entrances representing the real possible destinations inside a sector, routing tables of the actual nodes contain entrances belonging to all virtual nodes, except the one in the current sector.

The hierarchical routing system from [66] contains a Timetable updating system (TUS)

and a Route finding system (RFS) as principal components. Vehicles interact with RFS by sending information about the covered route together with the time spent covering the links and with TUS by sending their position. To bound the volume of communications an update interval of 30 seconds is imposed. The Route finding system uses a Hierarchical Ant Based Control algorithm. Different from previous ABC, three types of ants are used: *local ants*, *backward ants* and *exploring ants*. Local ants maintain routes between nodes of the same sector or routes to a virtual node if destination is in another sector. A local ant behaves similarly to a forward ant from ABC but is strictly limited to a sector. A backward ant follows the path of the local ant but in opposite direction while updating routing tables for the path and sub paths. Exploring ants on the other hand are responsible with recording the estimated time to travel between virtual nodes of sectors. Cars can be finally routed inside a sector following maximum values in the routing tables of probabilistic nature or if they have to travel to a another sector are routed via virtual nodes first.

By testing H-ABC in a realistic simulation environment with traffic-lights, round-abouts, precedence rules, multi-lane roads and 4 categories of vehicles, the authors of [66] claim to have developed a highly adaptive and dynamic routing algorithm. Furthermore, it was observed that a reduced to middle fragmentation of the network is sufficient.

Comparable to H-ABC, is the **City-Based Parking and Routing (CBPR)** system proposed in [9]. The similarity originates from the hierarchical routing that is analogous to the layered routing model of H-ABC, but also from the distributed multi-agent swarm intelligence approach used. An examination of CBPR reveals that intelligent lampposts which communicate with cars play the role of the Navigator. But at the same time an extra parking guidance capability dedicated to the city environment is introduced. It is claimed that this approach reduces traffic jams throughout the city and that increases traffic flow even when the number of the traffic participants is high. There is high degree of similarity between this approach and the H-ABC of Tatomir and Rothkrantz [66]. However, CBPR has two different types of exploring ants, that are: forward exploring and backward exploring ants.

Nevertheless, another intersection point can be established between BeeJamA, CBPR and H-ABC, namely the fact that the network has a hierarchical structure and wireless communication with the vehicles from traffic is required in each one of the algorithms. Table 2.2 offers a comparison in terms of type of layering, routing tables and type of agents used by these three hierarchical algorithms.

Certainly, the concept of hierarchical routing is quite widespread nowadays in wire-line telecommunication routing and in vehicle routing. Mainly, it is preferred due to several possible advantages as: it reduces the size of the routing tables, make networks more manageable and sometimes make the network metrics bounded and deterministic.

2.4 Traffic models for simulation of routing

Analysis and simulation complement each other. Analysis follows the simulation phase. It is not possible to simulate whole systems, but the main components can be determined and the basis for their interaction can be understood. An evaluation is needed also in the case of a routing algorithm/system for vehicular traffic. Introducing real time traffic information into transportation network routing makes it necessary to consider the formation of queues and traffic flows as a dynamic process.

Intriguing questions are: what is the average number of users in the system and what is the average delay induced by these users?

Table 2.2: Comparison between 3 hierarchical nature-inspired routing algorithms.

A parallelism between H-ABC, CBPR and BeeJamA is created by enumerating the layer types, the routing tables and the type of agents used in these swarm intelligence algorithms used for routing purposes.

Algorithm	**Layering**	**Routing tables**	**Types of agents**
H-ABC	*Local sector* *Global sector*	*Routing table*	*Local ants* *Backward ants*
CBPR		*Global routing table* *Local routing table*	*Forward Ant* *Backward Ant* *Forward Exploring Ant* *Backward Exploring Ant*
BeeJamA	*Net Layer* *Area Layer*	IFR_{net} IFZ_{net} FRM_{net} IFR_{area} IFZ_{area} FRM_{area}	*Short distance bee agents* *Long distance bee agents*

2.4.1 Simulating traffic flow

Traffic phenomena are complex interactions determined by a large number of vehicles. Due to human involvement, vehicles can form clusters when the density of a certain area is a lot above average. Traffic flow theory refers to the study of interactions between drivers, vehicles and infrastructure with the aim of understanding a network and avoiding traffic congestion problems.

The study of traffic flow is important because if traffic flow could be completely understood, then the traffic levels could be determined and would lead to congestion forecasting and avoidance. Measuring traffic flow can be realised in many ways: by conducting traffic surveys at specific locations or in an automatic manner using inductive loops sensors or other kind of devices. Recently, scientists are concerned with collecting reliable traffic data that has no temporal or spatial discontinuity. One of the generally suggested values of standard flow in case of optimal conditions is that of 2000 vehicles per hour per lane. However reality has proven that this flow can vary. In general, the term flow is used as a shorthand for rate of flow.

Nevertheless, only knowing the flow rate is insufficient to understand the phenomenon of traffic congestion formation. For example, a number of 10 vehicles could pass an inductive loop in a minute spaced at 60 *mph* or nose-to-tail in a traffic jam. The macroscopic traffic flow characteristic called density allows us to get an idea of how crowded a certain section of the road actually is. It usually expresses the number of vehicles in a certain amount of road. Vehicle density is measured in vehicles per km, per lane and requires the vehicle speed in addition to flow rate for calculation. On the downside, the concept of density totally ignores the effects of traffic composition and vehicles lengths as it only considers an abstract quantity, "the number of vehicles". The optimal density on a normal road should be of about 40 vehicles per km per lane. At this density the flow rate should be the one mentioned above. When density can not be computed or measured is it has to be estimated. While flow is a temporal measurement, density is considered a spatial measurement.

Traffic models

When studying traffic problems, like routing, it is highly important to use a proper traffic model. Traffic models for simulation can be classified based on:

- *physical interpretation*: **white box (deductive)**, **black box (inductive)** or **grey**

box (intermediate) models [5, 43]). In purely deductive approaches, accurate physical laws are applied. Whereas in black box or inductive models a parametrised model is fitted to the output of systems that measure traffic conditions. Last,in the intermediate models, are a combination of white box and black box models.

- *level of detail*: **macroscopic**, **microscopic**, **mesoscopic** [5]. Essentially, these models can be used to describe the evolution of traffic flows in the network. More precisely these models load the flows and propagate them from origin to destination inside the traffic network.

  1. A **macroscopic model** deals with averages of traffic states. In general, a macroscopic traffic simulator describes the evolution of macroscopic velocity and the vehicle density which are averages of the microscopic velocities of vehicles. Individual velocity variations and distance distributions are not considered. Usually, macroscopic models specifies the relationship that exists between: the traffic density, the average velocity and the traffic flow. These kinds of models describe large traffic networks in limited detail.

     Within this class, a further classification can be made that contains: *first-order traffic flow models* and *second-order traffic flow model*. The first one is based on the analogy of traffic flows with flows of fluids and is good at understanding: stop-and-go movements, but it is only based on traffic density. On the other hand, the second-order traffic flow model is concerned with traffic density and average velocity. There has also been a *third-order flow model* proposed by Helbing. This incorporates among the variables, the variance on the velocity.

     Many researchers believe that macroscopic traffic models fail to reproduce the essence of traffic. All stop and go waves of traffic or even congested traffic self-organized structures are regarded here as smooth.

  2. **Microscopic models** describe the complete dynamics of every single vehicle unit including all the variables that represent microscopic properties like position and velocity of the vehicle. A microscopic simulator explicitly represents a control strategy that mimics the movement of vehicles updating permanently their position. One advantage of being closer to reality is that infrastructure is incorporated directly. But the computational times of the microscopic model are also higher than that of a macroscopic simulator. The demand in the microscopic models is computed in two separate ways according to [14]. First method estimates the individual trips by an Origin-Destination matrix (O-D). Normally this patterns vary over time and there has to be a OD matrix for each period. The second method models the flow entering the network and the turning percentage at each intersection.

     Two important microscopic vehicle models are the *car following model* and the *lane changing model*. Microscopic vehicle models can also be implemented as *cellular automaton models*. A cellular automaton describes each highway as a network of interconnected cells. But even though cellular automaton models lack the accuracy of the time-continuous car-following models, they still have the capacity to reproduce a wide range of traffic related phenomena.

  3. **Mesoscopic model** uses a simplified node link-representation of the road network to track in an approximate manner vehicles' trajectories. Mesoscopic traffic models describe traffic in fewer details than microscopic traffic models do.

     There are different types of mesoscopic models such as *headway distribution models*, *cluster models* or *packet models* and *gas-kinetic models*.

- *representation of the processes*: **deterministic** versus **stochastic** [5, 43]. The stochastic traffic model implies at least one random variable that captures the variation in reaction time of the driver or the arrival processes. In this case, every simulation with the same initial conditions that is run twice will give different output. The former model (deterministic) has no random variables implying that all actors in the model are defined by exact relationships. The output of the model will be the same every time.

- *scale of the independent variables*: **discrete** versus **continuous** [5, 43]. A vehicle enters motorway stretches and travels through time and space to the end of them. Traffic models describe the evolution of traffic which depends of two variables: space and time. These variables can be discrete or continuous. Discrete models assume that traffic changes occur at discrete time instants. But also other variables, like position, can be assumed discrete. On the other hand, a continuous traffic model describes how traffic behaves over continuous time and in response to continuous stimuli. Nevertheless, the level of complexity for large networks can be overwhelming sometimes, and in consequence the traffic situations are discretezed before a simulation can be implemented.

- *operationalisation*: **analytical** versus **simulation** [5, 43]. Traffic models can be described in terms of numerical equations and then they are analytical or by reproducing the changes of the system over time and space and they constitute a simulation.

- **time-stepped** or **event-based** [14]. A time-stepped traffic model calculates the changes in the system in finite steps, while an event-based calculates the changes in the system when an event occurs.

A complete overview and classification of existing examples of traffic models can be found in [43].

2.5 Conclusion of literature overview

During the literature survey we found little methodological work focussing on evaluating and comparing the ability to adapt to traffic congestion of various models and algorithms. Dynamic network modelling has been a neglected relative of static network modelling. This is also partially due to the lack of data sets to study and due to the complicated nature of the problem. Some type of dynamic problems were covered by swarm intelligence algorithms: AS, ABC, H-ABC, AntNet, BeeHive, BeeJamA, CBPR. All of this mentioned algorithms result from the same bilogical inspiration but they apply different rules to construct solutions. However, something is missing from these algorithms: the integration of variable traffic delays. To cover this gap, we will focus on adapting the routing process to the variable travel time conditions that are found in real world road networks. Questions that come to our mind are:

- What is the appropriate cost estimation methodology for a network with time-dependent travel time?

- Do we actually have to simulate vehicle traffic flows?

- If we develop and algorithm that considers variable traffic delays, which is the best way to prove its efficiency?

As we have discovered in the literature review, swarm intelligence (especially ant based algorithms) provides *state of the art* techniques for routing, which have already

been proven efficient in telecommunication networks. On the downside, ant based algorithms usually require that each network node sends agents, periodically, to all the other nodes in the network. This is not necessarily ideal because when the number of nodes in the network increases, the algorithm processing time grows quadratically and the performance degenerates considerably. As we have seen, scalability issues, can be avoided by using a hierarchical network structure and routing alternatively in one of the layers that comprises this kind of structure.

Telecommunication networks are alike vehicular traffic networks due to the dynamic conditions that might arise in the traffic. We consider following the inspiration provided by the behaviour of ant colonies and implementing it as a method for our system of routing individual travellers, while avoiding traffic congestion.

The observed properties of ant colonies, namely: scalability, emergent and intelligent behaviour, autonomy, the stigmeric communication and the local information storage support the previously general design goal of routing travelers and creating routes that avoid congestion. Moreover, the high interest in insect societies can be understood by analyzing these systems. The presence of a set of distributed and autonomous agents characterised by local interactions that exhibit an adaptive behaviour to changes in the environment can be more than beneficial for routing in every day traffic conditions. Swarm intelligence posses great potential which should be used much more for routing drivers in a road network. Through this thesis we would like to accentuate the importance and the benefits swarm intelligence can bring to routing.

Chapter 3

Model Design

In this chapter we detail the model of the design, that we expect is going to satisfy the initial requirements. Later we will test our expectations.

If one remembers, our main goal is to design a flexible, robust traffic algorithm that will support self-interested agents in the context of dynamic network conditions. In other words drivers need to be routed reliable from a given source to a destination. Best routes have to be provided by an algorithm taking into account the traffic situation. Further on we detail the model we are going to adopt in several steps.

First, the general design possibilities for a routing system will be given in Section 3.1. These possibilities refer to cost of route segments computation and to the processing of the user request. Additionally, the first section also talks about the possibilities that one has in choosing the routing algorithm and details two types of technical system architecture (centralized or decentralized). Besides this, the first subsection will present reasons for choosing between a simulation or a numerical computation of the results. Output, or route representation is also one of the topics of the fist subsection.

In the second step of the model design, the actual resources required for the current routing algorithm are going to be described. A geographical infrastructure model and traffic measurements will be considered as inputs of the proposed algorithm. Route planning is essentially related to the transportation infrastructure and cannot be realised without modelling the actual infrastructure. The actual transportation network is going to be modelled as a **spatial-temporal network**. Moreover, we assume that real-time measurements of traffic variable constitute one of the main inputs of the algorithm that will provide an adaptive/dynamic fastest route in time for drivers. Travel-time (real-time or historical) over the entire road network are coming from a traffic provider, for example ANWB, which collects it from sensors along the roads.

Next, in the third step, we provide a formal description of the metaheuristic proposed which is comparable to that given in the paper where AntNet was first introduced by Di Caro, [23] and to ABC, introduced in [57, 58]. It follows from the characteristics of the routing problem and from the literature survey that it can be best solved with a multi-agent approach. AntNet and ABC, inspired from the previous work related to artificial ant colony techniques are such a paradigms. The original idea however comes from observing the behaviour of real ants related to food resource exploitation. In the previous part of this thesis, Section 2.3 emphasised the relationship to the most important frameworks of Swarm Intelligence and presented a descriptive working of ant-based algorithms.

Following the actual description of the algorithm, we revise in Section 3.4 and try to improve, the travel time computation for the routes. Last section of this chapter describes how vehicular traffic is going to be generated in this particular case.

3.1 General design aspects

This subsection is about general design possibilities or choices that are relevant for a routing system. The list may be longer, but these are the choices that we find of interest in the case of this project.

3.1.1 Centralized versus decentralized technical architecture

What kind of system architecture can we use? We have to discuss the technical framework within which the routing service is going to be developed and whether is going to be a centralized or decentralized one. Since this topic has been partially touched in Section 2.1, we have to decide which is the best course of action in the case of this project.

Conceptual decentralized system architecture

In a *decentralized routing architecture* the computation of the desired route takes place for each vehicle independently, on board of that vehicle (see Figure 3.1). It is ideal if vehicles receive traffic information via the wireless interface of their navigation devices. Information can be transmitted:

- via inter-vehicle communication of vehicle that are in a closed range,
- from a real-time/historical traffic database administered by a traffic management provider that gathers the information from a network of sensors (loop inductors, cameras, etc.),
- or directly from road sensors (the same as above).

Each vehicle in a decentralized architecture is supposed to have also a *Road Network Map* and a *GPS*, a spatial geographical positioning system.

Conceptual centralized system architecture

On the other hand, in a *centralized architecture system* (see Figure 3.2) a central server is responsible not only for monitoring the dynamics of the whole network but also responsible for computing routes in response to requests from the users.

The central server can have the following data inputs:

- a *spatial network model* of the actual *road network infrastructure*,
- *time delay profiles* or *kilometre delay profiles* based on a historical database, and that are coming from a traffic management provider which collects this kind of measurements (ANWB),
- or access to *real-time traffic information database*.

In turn, a user has the following options, depending of the circumstances:

- to forward route request via his on board navigation device with desired starting and destination position,
- or if he is the possession of cellular phone to forward only the route request, while a positioning system tracks its spatial position and forwards it to the central server.

Based on the previous data and user inputs, the central server will perform an algorithm that should provide directions for an optimal path and estimate the expected arrival time at the destination. We are not interested at this moment how the algorithm's processing happens.

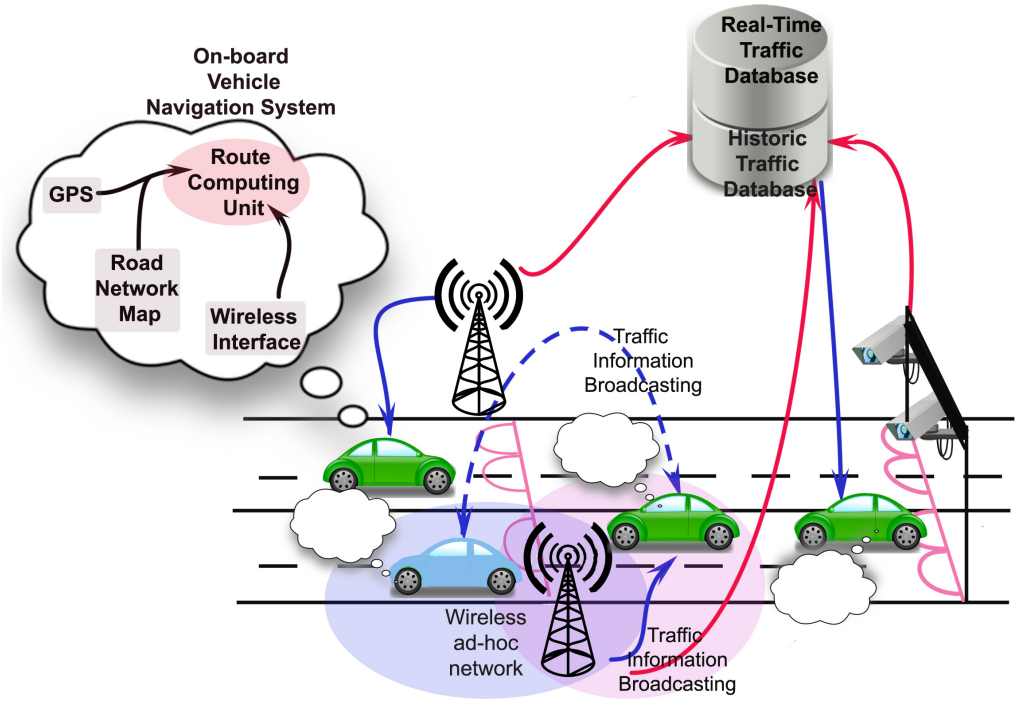


Figure 3.1: Technical infrastructure of decentralised routing system.

Vehicles have the capacity to compute their desired route based on the GPS position and speed in the Road Network Map. Via the wireless interface any vehicle receives traffic information either from sensors that are in a certain range, from a real-time/historical traffic database or from other vehicles. Traffic information is collected by sensors and inductor loops on the road or by surveillance cameras. Of course, the traffic management part can be more complex than in this figure but we are not interested in that.

Advantages and disadvantages of both architectures

In a centralised architecture the results can be analysed much easier. A central system is vulnerable in case the system's server breaks down. On the other hand, when evaluating a routing algorithm in a decentralized architecture, namely a mobile ad-hoc network (comprised of cell phones and handheld devices), the results are scarcely repeatable and reliable. Nevertheless, it involves real-time access to restricted traffic information databases for each user. Another possible disadvantage of the decentralized architecture is that traditional algorithms have to be modified in order to be used. What is more, a decentralized architecture involves some extra communication. The amount of information that has to be transferred to a part of the decentralized system is variable. Because of this factor a decentralized architecture can be faster or slower in processing user requests. One advantage brought by the decentralized system is: the possible increase in speed plus memory space. A decentralized system results in the diminution of the susceptibility to failure.

We plan to use a decentralized routing algorithm, which fits with a decentralized architecture. But the decentralized infrastructure is not yet available. The routing algorithm can be prepared to be distributed over several on-board vehicle navigation systems or computers (Figure 3.1), but at this moment can only executed in a centralized manner.

In the current purpose, the algorithm is part of a centralized decision support sys-

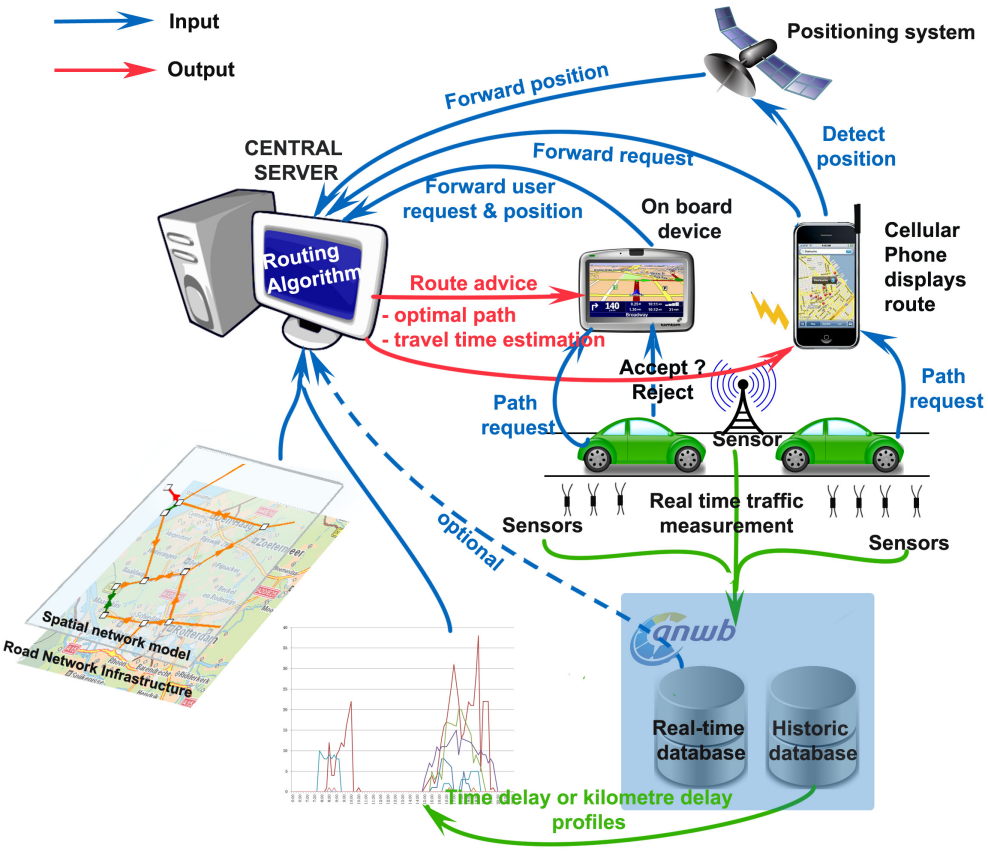


Figure 3.2: Technical architecture of centralised routing system.
Possible inputs and outputs underlying a routing system are also shown here.

tem which gives route advice to drivers who are equipped with an handheld devices, Figure 3.2. Moreover we suppose that the decision support system that gives the route output can detect the spatial position of these travellers via a positioning system. Drivers can accept or reject the advice given by the routing system.

3.1.2 Travel time models

The routing model should consider all route alternatives available in the network and should determine the route that provides the optimal route cost taking into account actual and future traffic conditions.

Next, because we want to compute the shortest route in time between two points in a selected network, we are going to consider the cost of a route as the travelling time. The term cost and traveling time will be used alternately.

Let c^r_{ijt} denote the cost of the route r between two intersections i and j when departing at time t, then the minimal cost route is represented by:

$$min\ c^r_{ijt}. \tag{3.1}$$

Normally, the route costs are computed by a summation of the corresponding costs of each edge in the route. But the question is how to determine the costs c^r_{ijt}.

Traveling time estimation in instantaneous and actual route choice

The cost can be considered in two ways ([68]) depending on the kind of route choice one would like to model:

- **instantaneous route choice**

  Traffic modelling tends to be a lot "backward-looking". Among the main assumptions that are made in traffic modelling, one says that the present or future traffic condition is similar to the traffic condition from a couple of minutes ago. Travellers assume that the link travel costs (link travel times) of the entire network will not change from the departure time until they end a trip along a specific route. Put differently, they compute the instantaneous route costs by:

  $$c_{ijt}^{r} = \sum_{a \in r} c_{at}, \tag{3.2}$$

  and chose the route r that yields the lowest cost. The previous equation (3.2) is the sum of the cost c_{at} of each link a at time instant t when departing at the same instant.

  There can be times when this supposition is valid, for example, during the night time when there are fewer vehicles on the road an the minimum travel time is the actual travel time. But at the start of the rush hours it fails to be true anymore. Systems that consider this are not efficient in reacting to changes in capacity demand and don't help too much in relieving the congestion.

- **actual (ideal) route choice**

  In this route choice model, travellers regard the link travel costs of the network as the actual travel times they will experience when making a trip. Link cost may vary over time, so the decision of the route choice is better in this case than in the instantaneous case. It is believed that by considering the actual travel time, there aren't any alternative routes that would be better when travellers have reached their destination. However this requires long time planning: travellers know from experience the best route alternative. Even though actual route choice modelling generates more realistic results, for the long term, it is more difficult to accomplish because the route costs are not computed as easily as in equation (3.2).

  The actual costs are computed in the following way:

  $$c_{ijt}^{r} = \sum_{a \in r} c_{a\theta_a}, \tag{3.3}$$

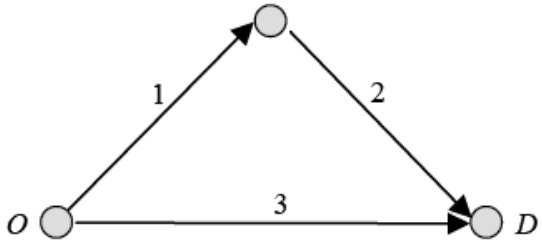
  where $c_{a\theta}$ is the cost at θ_a time instant when a vehicle reaches link a if it departed at time t from i heading towards j. This value can be defined according to:

  $$\theta_a = \begin{cases} t, \text{ if link } a \text{ is the first link on route } r \\ \theta_{a^-} + t_{a\theta_{a^-}}, \text{ otherwise,} \end{cases} \tag{3.4}$$

where a^- is the previous link of a on route r and $t_{ak} = t_{ij}(\mathcal{I}_k)$ is the travel time on link a at time k. Underlying this route choice model is the assumption that we can deal with time as a continuous attribute.

The traveling time resulting from a instantaneous route choice will be further on called instantaneous cost/traveling time and the traveling time resulting from an actual route choice will be called actual cost/traveling time.

Difference between instantaneous and actual route choice



τ_{ak}	$k=1$	$k=2$
$a=1$	1	2
$a=2$	1	3
$a=3$	3	3

Figure 3.3: Network for instantaneous and actual route choice calculation.

Variable a represents the edge, while variable k represents the time instant. For 2 time instants $k=1$ and $k=2$ the cost of each edge a is given in the table.

If one considers the simplified network in Figure 3.3 there are two routes that have as source node O and as destination node D. The second route $R_2 = (3)$ (see definition given at page xi) is using only link 3 (a direct route) while the first route $R_1 = (1, 2)$ is using links 1 and 2. Dynamic travel times are given in the table near the same figure for time instant $k = 1$ and $k = 2$.

When departing at time $k = 1$ the instantaneous route costs are:

- for R_1: $t_{11} + t_{21} = 1 + 1 = 2$,
- for R_2: $t_{31} = 3$.

When departing at time $k = 1$ the actual route costs are:

- for R_1: $t_{11} + t_{21+t_{11}} = 1 + 3 = 4$,
- for R_2: $t_{31} = 3$.

If a traveller would consider the instantaneous route cost, the choice will be route 1. Otherwise, if a traveller would base his decision on the actual route costs, he would choose route 2, since in that case the actually experienced route costs are minimal. In the actual route choice model,however, we need to know the dynamic link travel times or at least have an estimation of this time.

Travel time estimation from acquired data versus prediction

We have previously seen that the actual cost of a route is computed based on instantaneous cost of the same route. To estimate the instantaneous cost of a network, this has to be done for every segment in that network. Usually, this is accomplished based on the following formula:

$$t_{ij} = \frac{d(i,j)}{v_{ij}} \tag{3.5}$$

But the instantaneous average speed on every segment $e_{ij} = (i, j)$ and the length of each segment $d(i, j)$ have to be previously available. If we consider that the distance of the roads segments is measurable, then v_{ij} is the only variable missing. The speed can come from historical data, namely from acquired data. Theoretically, if samples from already gathered data consisting of average speeds of traffic flow by road segment are considered, the instantaneous traveling time cost could be estimated for an entire network. Such a sample of average speeds can be produced for time segments as short as 5 minutes, for every hour, day, season and even for a special season. Moreover a sample of data can gather the averages of several historical days.

Because empirical evidence has shown that traffic patterns are changing from day to day or from season to season, this above, is the simple solution to the instantaneous travel time estimation problem. It is believed that travel speed prediction or travel time prediction can be successfully realised if an analysing of historical data is performed and profiles of these variables per road segments are created. On these profiles, *Bayesian Network analysis* with contextual metadata information or *Support Vector Machine* algorithms could be later applied. Bayesian analysis of the whole network speeds can result in detecting empirical correlations between the *network variables* (speeds, road closures, etc.), *temporal variables* (calendar, holidays etc.) and *non-network variables* (weather, special events etc.). Empirical evidence has again proven that the impact of these three types of variables differs over time periods. Moreover, a disadvantage for predictions on large scale is that a high volume of data input needs to be handled. In practice, because enormous computing power is required, only limited coverage over time, for predictions is offered. Full coverage comes at a high cost. Traffic predictions are manageable in small pilot networks over a limited time interval but are very different for full networks with high detail of historical measurements.

3.1.3 Choosing the algorithm

In Chapter 2, the main idea in one of the most well-known routing algorithms (Dijkstra's algorithm) it was summarised together with the current state of the art techniques in the field of routing (swarm intelligence). Of course, the choice of the algorithm is crucial if we want to route drivers in the best way possible from a source to a destination so that instantaneous travel cost is considered. Nevertheless, the chosen algorithm should be appropriate also in relationship with the technical architecture selected.

From the insight gained from the literature review, ABC and AntNet are the choices that are quite appealing at this moment. On the other hand, neither these routing algorithms neither others have incorporated actual travel time computation in their structure. The optimisations of these algorithms considered in a greater measure how can they be applied to hierarchical network structure or which are the optimal parameters, but never how travel time is modelled. The routing process needs to be temporal sensitive, that is to be aware of the evolution of traffic in time.

Further on, in one of the next sections, we will explain how actual travel time computation is going to be integrated into ABC.

3.1.4 Analytical-based model versus simulation-based model

We feel obliged to discuss how the traffic model can be made operational. The literature survey has taught us that traffic models are usually put into practice either as an *analytical-based model* or as a *simulation model*. In the first case, the traffic system is described in terms of numerical equations, while in the second case changes of traffic over time are approximated in the model. Simulation models follow the dynamics of the traffic system. Drew defines a simulation ([34]) as a dynamic representation of some part of the real world achieved by building a computer model and moving it trough time. In general, traffic systems are an excellent application environment for simulation based research, an application area where the use of analytical tools is restricted to subproblem level. Because we do not dispose of many of the variables of traffic, the simulation is chosen in the case of our system.

Traffic systems are characterised by a number of features that are hard to analyse, to control and to optimise. The systems cover large physical areas and the number of participants is high, the objectives of the participants are not necessarily identical with each other or identical to those of the system operator (SO vs UE) and there are many inputs that contribute to the traffic conditions. We propose here a model that optimizes

the route of individuals, so it *concerns only UE and not SO*. To implement an algorithm that specialised in System Equilibrium, traffic demand, traffic flow have to be known. The traffic assignment problem in that case has to be also solved. Even though the simulation of travel demand field has grown considerably in the recent years, this is not the scope of the current project.

Any road network is inherently dynamic by nature, that is the number of vehicles and the speed varies in correspondence with time, with a high degree of randomness. A lot of simultaneous interactions are performed by all participants present at the same time in the system. If we were to model this interactions we need to choose a microscopic traffic simulator, but instead we have chosen to have a type of *mezzoscopic* simulation.

3.1.5 Output representation and update management

Output representation. Last but not least from the general design aspects, is the the output representation. In an ideal system drivers need to have the route represented in a multimodal User Interface context, with voice, visual display on a map and text available to them. Implementing all of this options requires expertise in many sections of computer science and may require the work of several people over a longer period of time. The maps display of a route is however quite easy to implement and is feasible for our project.

Update management. Traffic situation may change after a route is sent to the user who asked for it. Updates can solve this problem, but when should be such an update sent? Should the updates be sent only on request, or without any new request from the user? It is difficult to answer this question since preferences of the users may vary in this respect. User tests, or modeling user's profile of requests are directions that can be followed in this perspective. But have we already mentioned in Section 1.2.3, that modeling user preferences is not part of this project.

In this project, the routing advice offered to travelers is going to be en-trip, on an intersection to intersection basis.

3.2 Resources for the current routing system

Designing or improving a routing algorithm will not help anyone unless it is considered also in a realistic scenario, where its pluses and minuses can be counted. Finding shortest path between a pairs of physical locations is a complicated task, mainly because the travel times on road segments depend on the time of the day. Road networks need to be modelled as spatio-temporal networks to account for the time-dependency. The real situation can be difficult to accomplish, but nonetheless there is always the option of testing it in an environment that is as close as possible to the reality. Such an environment can be described in terms of:

- a **spatial network model**, $G = (V, E, L)$, which enables the modeling of important aspects of road-network infrastructure. Different aspects of the road-network infrastructure are of interest: roads characteristics and junctions properties.

- a **temporal network model**, T, that should reflect the **evolution of traffic variables** as a time series. The traffic variables can be time delays/kilometres or the actual travel speed, on segments of the aforementioned spatial network graph, during a definite time period. These variables can be considered as the chaining weights of spatial network model (road network data model).

3.2.1 Spatial network formulation

One of the crucial aspects involved in the design of a vehicular traffic simulator is the representation of the actual road network infrastructure. A ***spatial network model*** (synonymous to *road network data model* or *transportation network model*) is essential for route-based queries or for finding alternative routes to a specific destination. Route-finding queries typically deal with determining the shortest route to a given destination.

The level of detail reproduced by the spatial network model: how are roads going to be encapsulated or how are turning sections (segments where drivers can change roads) going to be represented, has to be carefully weighted.

Like many other simulator network models, the one that we used is a simplified ***node-link*** representation that contains several basic elements. A node-link data model has the aim of capturing the topology of a road network by abstracting geographical detail. The representations of road networks obtained in this way are compact enough and very well suited for route finding queries (retrieval of unobstructed routes that satisfy shortest time criteria). Typically, a link-node model captures a road network as a collection of nodes connected by directed links. A road network may be modelled by several link-node instances that capture the road network at different levels of detail. For example a region may be modelled by two link-node instances: a very detailed one (low-level link-node model) and a higher-level link-node instance. This method has been adopted in [9, 72] as mentioned in Chapter 2. We stick to the low-level representation that matches real world network topology. In principle, the whole network consists of *locations*, *intersections* and *links*.

In **formal terms**, *the spatial network is defined as a directed graph* $G = (V, E, L)$ (see page xi), *that consists of:*

1. *a set of* ***intersections*** *or* ***nodes*** $V = \{1, \ldots, i, \ldots N\}$,
2. *a set of directed* ***edges*** *or* ***links*** $E = \{e_{ij} \mid e_{ij} = (i, j),\ i, j \in V\}$ *and* $|E| < \frac{n(n-1)}{2}$, *An edge or a link is defined as a pair of nodes* $e = (i, j),\ i, j \in V$, *distinguishing* ***start nodes*** *for a link* $f : E \to V,\ f = i$ *and* ***end nodes*** $l : E \to V,\ l = j$. *In addition, we consider that links have attached a certain cost* $w : E \to \mathbb{R}^+$. *However for an edge* $e_{ij} = (i, j)$ *and an edge that is going from* j *to* i: $e_{ji} = (j, i)$ *the cost* $w(e_{ij}) \neq w(e_{ji})$. *Links* e_{ij}, e_{ji} *are called* ***consecutive*** *if* $l(e_{ij}) = f(e_{ji})$. *All links for which* $e_{id} = (i, d),\ d \in V$ *are called* ***outgoing links*** *of node* i. *All links for which* $e_{di} = (d, i),\ d \in V$ *are called* ***incoming links*** *of node* i. *For every outgoing link there is an incoming link.*
3. *and a set of* ***locations*** $L = \{L_1, \ldots, L_i, \ldots L_N |\ 1 \leq i \leq N\}$ *where* $L_i = \{l_1, \ldots, l_r\}$, *is the set of locations for intersection* i *and* l_r *is a location.*

Given the previous graph, *a* ***path*** *or a* ***route*** R *from node* $s = f(e_{i_1 i_2})$ *to node* $d = l(e_{i_{p-1} i_p})$, *is a sequence of pairwise consecutive edges* $R = (e_{i_1 i_2}, e_{i_2 i_3} \ldots, e_{i_{p-1} i_p})$. An optimal path has minimal costs between source and destination.

An illustration of the definition is given in Figure 3.4. Table 3.1 enumerates the outgoing and incoming links that can be visualised in Figure 3.4.

The explanation of the components of the spatial network model in **informal words** is given next.

In this case, an ***intersection*** or a ***junction*** is a connection between two links that is constituted of several on/off ramps (entry/exit points of a motorway) called ***locations***. Figure 3.5 displays how the locations of intersection Amstel are situated geographically. Each intersection can be the source or the destination of a vehicle. Also, the source-intersections are places where vehicles are added to the network in cases of traffic simulation. While, on the other end, in the destination-intersections, the cars that have

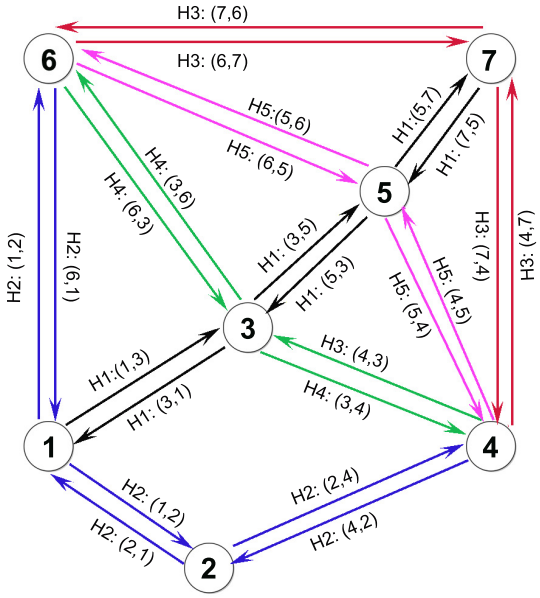


Figure 3.4: Potential graph of the node-link network.

The network is formed of several highways: H1,H2,H3,H4, H5, each one displayed with a different colour. A highway is divided into directional segments/links depicted trough arrows. Between two nodes i, j there are two links (i, j) and (j, i). The locations of every intersection are not represented here.

Table 3.1: List of outgoing and incoming links for each intersection in Figure 3.4.

Intersection i	Outgoing links from i	Incoming links to i
1	$(1,6)$, $(1,3)$, $(1,2)$	$(2,1)$, $(3,1)$, $(6,1)$
2	$(2,1)$, $(2,3)$, $(2,4)$	$(1,2)$, $(3,2)$, $(4,2)$
3	$(3,1)$, $(3,2)$, $(3,4)$, $(3,5)$, $(3,6)$	$(1,3)$, $(2,3)$, $(4,3)$, $(5,3)$, $(6,3)$
4	$(4,2)$, $(4,3)$, $(4,5)$, $(4,7)$	$(2,4)$, $(3,4)$, $(5,4)$, $(7,4)$
5	$(5,3)$, $(5,4)$, $(5,6)$, $(5,7)$	$(3,5)$, $(4,5)$, $(6,5)$, $(7,5)$
6	$(6,1)$, $(6,3)$, $(6,5)$, $(6,7)$	$(1,6)$, $(3,6)$, $(5,6)$, $(7,6)$
7	$(7,4)$, $(7,5)$, $(7,6)$	$(4,7)$, $(5,7)$, $(6,7)$

reached this point are simply removed each time step from the simulation. Intersections are identified by their names and their corresponding locations. The physical position of an intersection in the network is computed based on the physical positions of the locations associated with it. A route is formed from consecutive pairs of edges. An edge can only be changed at junctions.

Links are directional elements that connect intersections and correspond to the real highways, having mostly the same properties as them. Choosing highways instead of normal roads inside a city is motivated by the fact that traffic information is mostly available for highways. Most of the public traffic providers (for example in The Netherlands: ANWB `http://www.anwb.nl/verkeer`, `http://www.trafficnet.nl/`, De VerkeersInformatie Dienst `http://www.vid.nl/VI/overzicht`, `http://www.traphic.nl`, etc.) display real-traffic measurements mainly for highways and less for roads belonging to a city area. The traffic measurements offered by providers are minutes or kilometres of travel time delay.

On top of the previous advantage, highways are preferred because they don't contain any signalised intersections. Usually, the simulation of traffic and the routing process

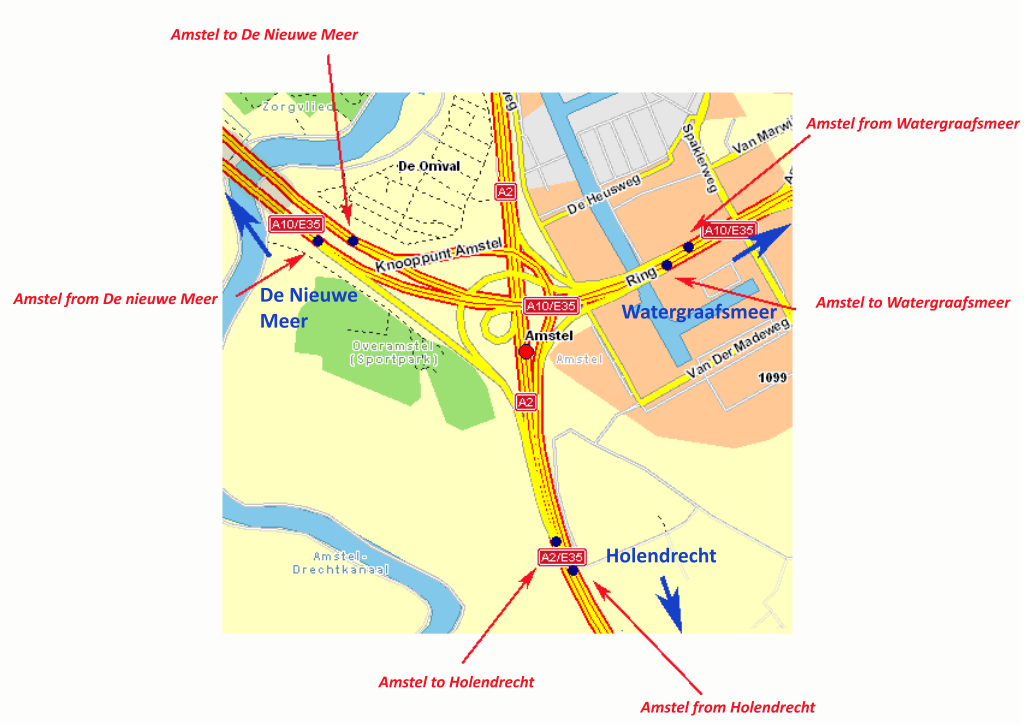


Figure 3.5: Geographical positioning of the locations of intersection Amstel.

The red circle is intersection Amstel. The blue circles are the locations that belong to this intersection. The locations represent the on/off ramps that are found at the crossing of A10/E35 with A2/E35, near Amsterdam.

are becoming more complex when traffic lights exist. Because in a city environment traffic information is not generally available, and because roads with traffic lights cannot be excluded, the routing process in this project aims only to cover highways. If one remembers, this was a limitation of the scope introduced in Chapter 1. We have imposed the constraint that only unsignalized intersections will be covered because when signalised intersections are considered, the number of tools for network analysis is quite small in comparison to programs for isolated intersections and road sections. Route computation usually implies that the search is done in the entire network and not only in isolated parts of the network.

In the current model, for a link, the internal parameters are *maximum speed*, *direction*, *length* and *capacity*.

To capture a road network in a format that is appropriate for low-level navigation, a simple directed graph is not always the most appropriate method. A route followed by a vehicle may be abstracted as a sequence of segments. But cost are not only costs of edges,they can be also identified as costs that appear when leaving an edge and entering the next. Vehicles are allowed to make turns when changing segments/links in the network or when changing lanes. To model this one can introduce *lanes* and *turns*.

Lane modeling

In some road network models the basic entities are lanes and a link is the aggregation of several lanes. In the context of routing, lane modeling is not only a finer level of geographic detail but it conceals important traffic flow details that might cause delays. If we remember from Chapter 1, one of the causes of traffic jam is the infrastructure layout. More specifically, lanes merging can affect the traffic flow on a road. It is good

to know that several lanes can merge not only in the vicinity of an intersection, but also along a link. Recently, there were several authors that recognized the importance of using lane modeling in transportation applications [19].

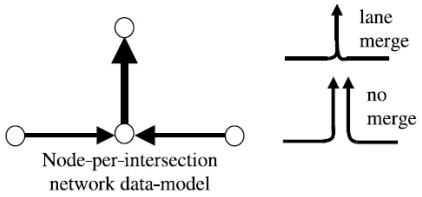


Figure 3.6: A node-per-intersection network flow data model.

This type of model cannot represent the difference between lane merge and non-merge.

Figure 3.6 shows how a flow data model in a standard *node-per-intersection network* can be interpreted in different ways. The upper and the lower case are indistinguishable according to this model. Nevertheless, this makes a great difference because in the upper case, the potential traffic delay may be bigger under high travel demand. This mainly happens because traffic is merging into a single lane. However, including them leads to the increase of routing ambiguity since an intersection might be in several states.

Congestion and suboptimal routing has to be avoided also in case delay is caused by merging lanes. A *node-per-lane model* can be used. We have already made a step forward towards this kind of representation by introducing locations that represent exit points or entries on a specific link.

Turn modeling

Turns or *transfer links* are synonyms for merging areas between different links in an intersection and need to be introduced to transfer cars between links. At the end of an edge a traveller makes a selection among continuing edges before proceeding with another link. Turning movement delays can be significant in the case of congested streets and ignoring them can lead to suboptimal paths. The cost of a turn cannot be stored as attributes of nodes and edges in the graph and they cannot be handled appropriately by shortest path algorithms without modifications. In general, turns can be stored in extra tables or their cost can be calculated on the fly from geometry.

When costs are not calculated on the fly, turns have been modelled using a turn table. Turns restrictions are represented as a row in the table that references the two associated edges. One other proposed method in literature to represent turn restrictions is node and graph expansion [73]. In this case, an intersection is expanded to a subgraph where permissible edges are depicted by edges. Turn costs are then represented by new edges of a *pseudo-dual graph* and the shortest path algorithm can run without modifications. However, including them leads to the increase lead to a substantial increase in the size of the network which adversely affects the performance. Nevertheless, the pseudo-dual graph is conceptually cleaner.

Even though turn modelling can prove to be good for the optimality of the routes provided by a routing algorithm, traffic information suppliers don't offer data for these part of the highways. In consequence, in our model we calculate the costs (travel time) of the turns on the fly from the length and the maximum speed of the respective road.

3.2.2 Temporal network formulation

Travel times on road segments, very often dependant of the time of the day due to varying levels of congestion, are making the shortest path also time-dependant. Road networks need to be modelled as spatio-temporal networks to account for this time-dependence.

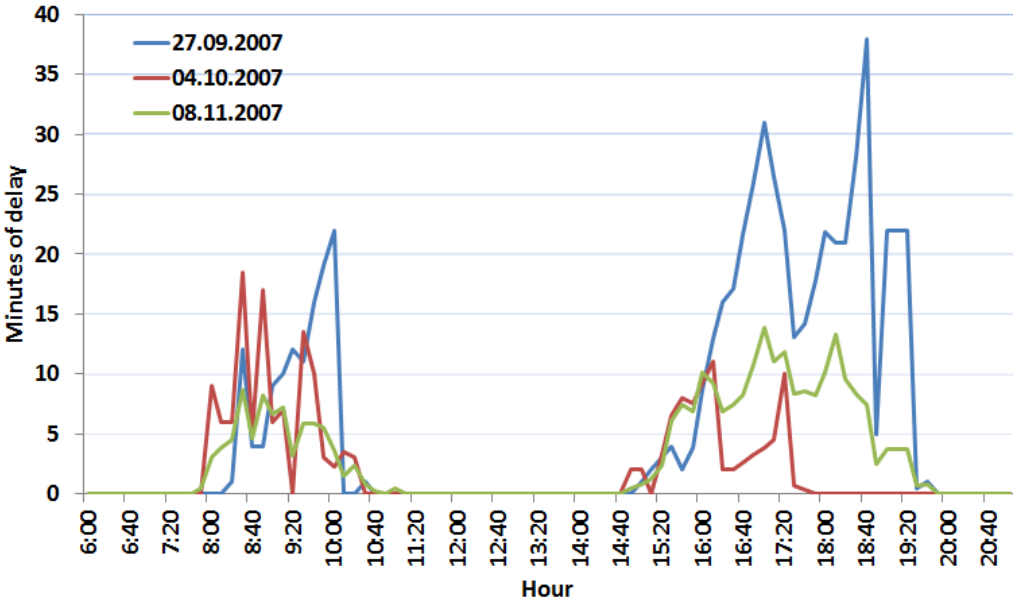


Figure 3.7: Delay distribution for the segment Holendrecht Oudenrijn from A2.

The data is coming from the ANWB website and is gathered for 3 days: 27.09.2007, 04.10.2007 and 08.11.2007.

Motivated by applications of road traffic control, we acknowledge that transit times on the arcs of a spatial network model may change over time due to car flow variation. But one thing is certain even from empirical evidence, and that is: flow varies with time. Figure 3.7 resulting from empirical observations is to prove that travel time is evolving during a day time interval. Moreover, the traffic delay is different from day to day on the same road. The delay that drivers encounter over the normal travel time is depicted there. One can distinguish 2 significant intervals for the delay: a *morning peak* between 7:30 and 11:00 and an *evening peak* from 15:00 to 19:30.

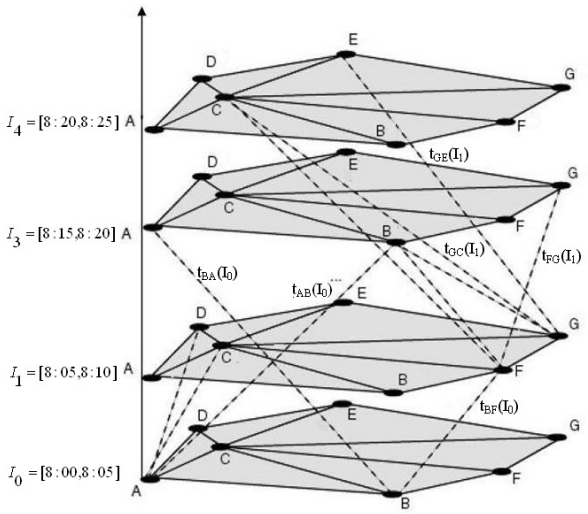


Figure 3.8: A three-dimensional time-expanded model of the costs of a road network [35].

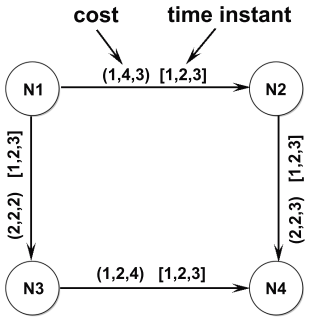


Figure 3.9: Time aggregated graph of the costs of a road network.

In this context, both ***time-expanded networks*** [45], and ***time-aggregated graphs*** [40], are being explored in literature. Time expanded networks represent the time dependence by copying the *spatial network graph* for every time instant, as in Figure 3.8. Time is divided in intervals on the z-axis. The actual costs between vertices is represented by dashed line between planes. These lines connect the copies of the nodes at various time instants. The solid lines represent the connectivity of the nodes in the network, the actual edges on which simulated cars can travel. In time aggregated graphs (see example of Figure 3.9) the time-varying attributes are aggregated over edges and nodes. Figure 3.9 shows a network at three time instants for which the edges' weights change over time. An edge has two attributes, both time series. The first attribute from left represents the weight of the edge and the second the specific time instant. For example, the cost of the edge $e_{N_1N_2} = (N1, N2)$ changes from 1 at $t = 1$ to 4 at $t = 2$ and it reaches 3 at $t = 3$.

Needless to say, the instantaneous and the actual route choice (see Figure 3.3) require a spatial and a temporal network model in order to be applied. In this thesis, we follow the approach of time-extended network.

Because the travelling times on roads (in the considered spatial network) are not constant over time we subdivide the time into subintervals. By adopting this method, travel time can be satisfactory approximated for each subinterval. For a determined time horizon $\mathcal{T}$, discrete time intervals $\mathcal{I}_k = [start_k, end_k]$ are constructed in accordance with the travel time delays that are considered.

3.3 Algorithmic approach

Below we describe how AntNet is going to be extended and improved for vehicular traffic routing. What follows is a detailed description of the main characteristics of the algorithm that is going to be implemented together with the central data structure used, the forward and the backward agents. We leave at the end, the extension that we propose for AntNet. For now, it sufficient to know that it is has to do with travel time estimation methodology.

The algorithm that we propose for solving the goals stated in Section 1.2.2 behaves in a similar manner to that of AntNet [23]. But, to this approach we have added several modifications through which we expect the routing process will benefit.

The nucleus of ant colony techniques is comprised of a parallel search for the best solution realised by a colony of artificial agents called "ants". AntNet can be best described in terms of two artificial deliberative agents: forward and backward ants. Both types have the same structure but their functionality is different. These types of agents communicate in an indirect way according to the stigmergy paradigm, as stated in Section 2.3.1. AntNet makes use of the pheromone feedback from both forward and backward ants. By concurrent reading and writing from pheromone tables present at each intersection/node of the network, information that will be further used for routing is accumulated.

From the point of view of the agents, AntNet is a *distributed/decentralised multipath* routing algorithm that uses *online delayed pheromone update* for all minimum delay paths and subpaths found by ant agents. AntNet can be described as composed of a solution construction phase and a data structure update phase. The fluctuations of the network traffic can substantially affect the learning process that ants are part of. But a balance has to be found so that the routes chosen are quite stable.

3.3.1 Description of an example

Given the network in Figure 3.10 we present here an example of the working of the simple ant-like agents that constitute the core of our algorithm.

A forward ant $\mathcal{F}_{AD}$ that leaves source A in order to discover a feasible path towards the destination D, will move from intersection to intersection based on a probabilistic

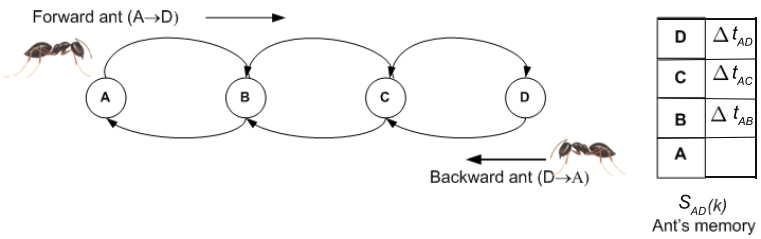


Figure 3.10: Example of ant behaviour.

Any forward ant $\mathcal{F}_{AD}$ will move along the path $A \to B \to C \to D$ and once arrived at node D will launch the backward ant $\mathcal{B}_{DA}$ that will move in the opposite direction. The memory of the forward and backward ant is represented by $S_{AD}(k)$.

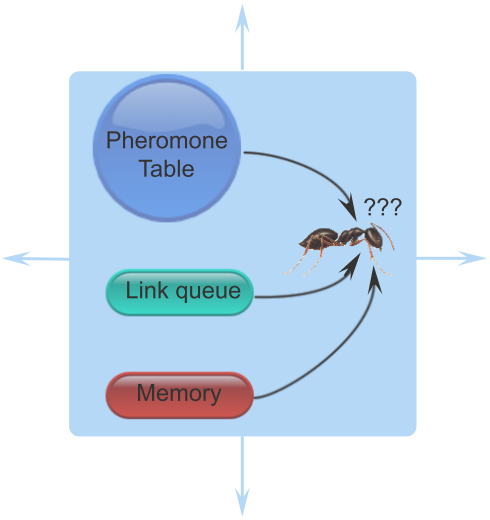


Figure 3.11: Decision factors of the forward ant at each intersection.

In general, the decision policy of the forward ant is parametrised by the pheromone table, by the status of the link queue (heuristic information) and by the memory of the already visited nodes (counts for loop avoidance in exploration).

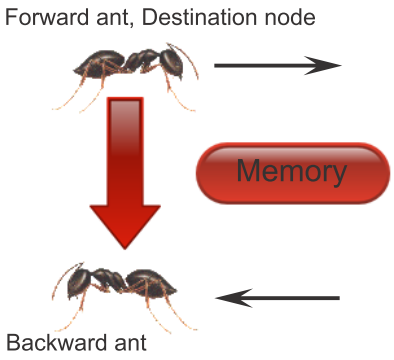


Figure 3.12: The forward ant creates the backward ant at the destination.

The backward inherits from the forward ant all its memory.

decision rule. The decision policy (Figure 3.11), depends mainly on the probability table encountered at each reached node, but also depends on memory of the ant and on the link queue at that node. It is not guaranteed that the forward ant $\mathcal{F}_{AD}$ will reach its destination since it may run in circles, but the ant can use the memory that is available to her to avoid running in an infinite loop. Lets assume that the forward ant reaches D and moves along the path $A \to B \to C \to D$, Figure 3.13. Once arrived in destination D, the forward ant will create a corresponding backward ant (Figure 3.12) that is traversing the reversed road $D \to C \to B \to A$ and will die. The forward ant stacks into memory, during her trip, the name of the current intersection and the travelling time t_{id} that has passed from leaving the source. The computed travelling time t_{id} is used by the backward ant to alter the average travelling time μ_d for the destination of the current route. This average travel time determines how optimal a route really is.

When arriving at the destination intersection, an "optimal" route between source and destination is known to exist. Not only the route between the source and destination is

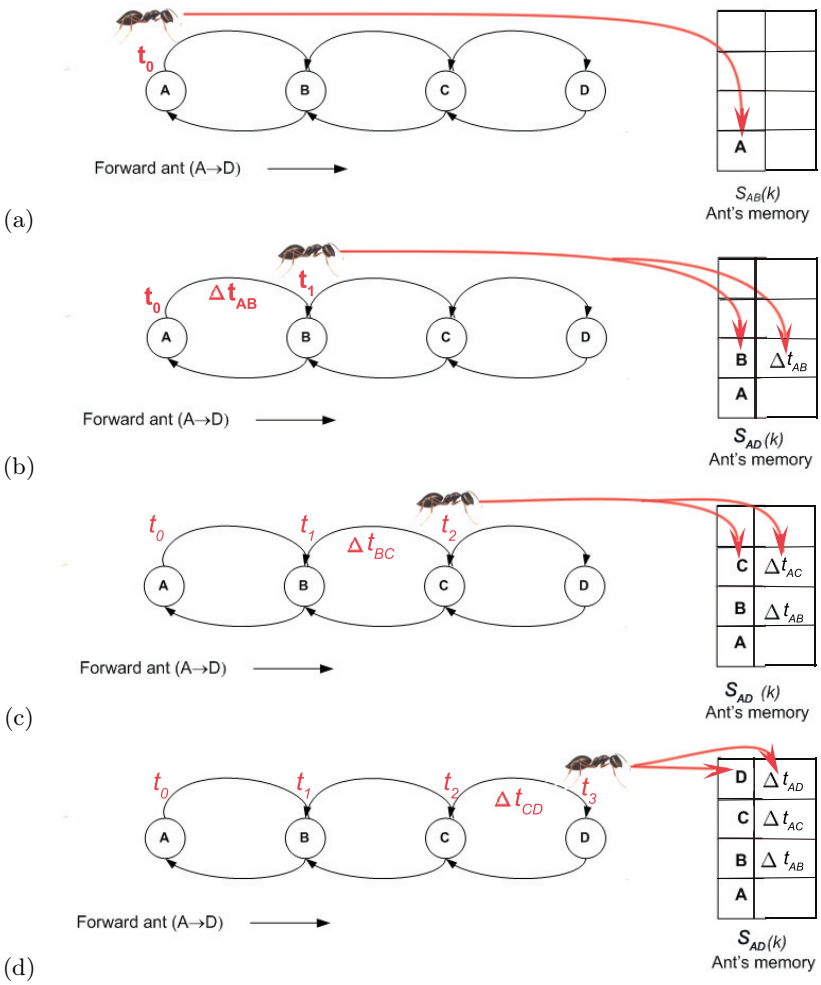


Figure 3.13: Detailed example forward ant behaviour.

The forward ant $\mathcal{F}_{AD}$ starts in source $s = A$ and moves from intersection to intersection, stores every current node $i, i \in \{A, B, C, D\}$ in the stack $S_{AD}(k)$ together with the arrival time t_{Ai}.

optimal but also between each intermediate intersection towards the destination intersection is optimal. The probabilities stored in the routing table of each node can now be altered to reflect the information gathered by the forward ant in the stack. Because the forward ant stacks its entire memory in the stack $S_{AD}(k)$, the backward is able to retrace the path and to update the probabilities.

At every node $k = C, B, A$ the backward ant $\mathcal{B}_{DA}$ will use the stack $S_{AB}(k)$ to update the pheromone values:

- When in node D, Figure 3.14.a, the traveling time t_{AD} is popped out and used for alternating the previous average travel time μ_D.
- When in node C, Figure 3.14.b, the travel time t_{AC} is used to update μ_C. Moreover, the pheromone value corresponding to the entry $\mathcal{P}_{DD}$ will be **incremented**. The probability values for the rest of the neighbours will be decremented. But since in this graph the intersection C has only two neighbours: B and D, the probability $\mathcal{P}_{DB}$ will be **decremented**.
- When the backward ant arrives in node B, Figure 3.14.c, the average travel time

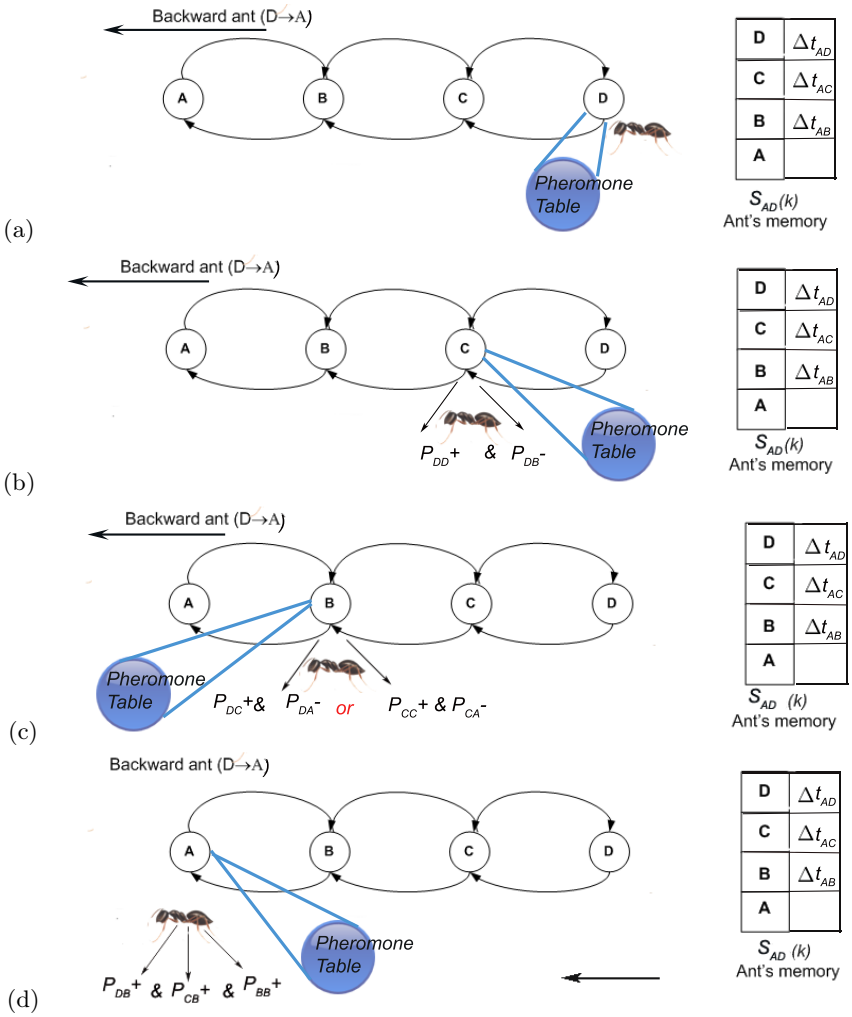


Figure 3.14: Detailed example of backward ant behaviour.

Backward ant $\mathcal{B}_{DA}$ leaving from $d = D$ towards $s = A$ retraces the path of forward ant $\mathcal{F}_{AD}$. Nodes from stack $S_{AD}(k)$ fare used to update probabilities. Traveling times popped from memory are used to update the average travelling time to every node $i, i \in \{A, B, C, D\}$ in the selected path.

μ_B gets altered. Also, the probability values corresponding to the path towards the destination and all possible subpaths will be updated. For the path towards destination, $\mathcal{P}_{DC}$ will be **incremented** and $\mathcal{P}_{DA}$ will be **decremented**. There is only one subpath from B in the route towards destination D. For this, the probability value $\mathcal{P}_{CC}$ will be **incremented** and the probability value $\mathcal{P}_{CA}$ will be **decremented**.

- When in node A, Figure 3.14.d, the backward ant will update the average travel time μ_A. Besides, the backward ant will **increment** the probability values: $\mathcal{P}_{DA}$, $\mathcal{P}_{CA}$, $\mathcal{P}_{BA}$.

Figure 3.15 exemplifies in a visual form which probabilities are updated for the path and the subpaths. As one can see, each backward ant does $n(n-1)/2$ updates of probabilities, where n is the number of nodes in the path. So the growth of updates grows quadratic with the number of nodes in the path.

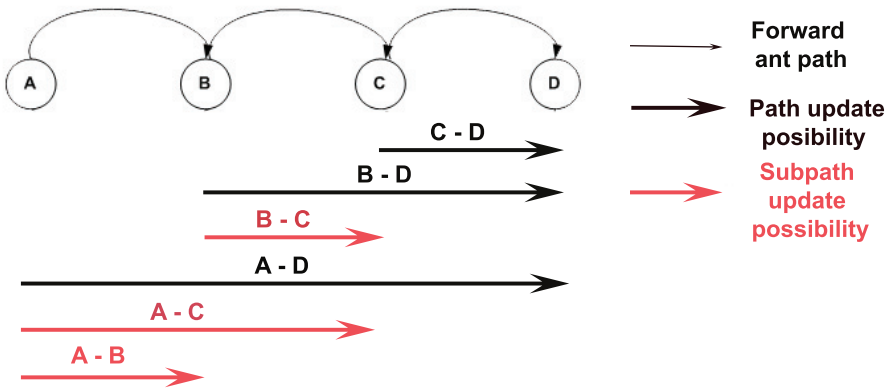


Figure 3.15: Example of probabilities updated by backward ant.

3.3.2 Routing table

Routing algorithms usually keep track of the status of a network through data structures that belong to each network node, namely routing tables. A routing table $\mathcal{R}_i$ is a structure of probabilistic entries in our case (see Figure 3.16). The routing table values are the result of continuous learning process and capture the past, the present and we expect, the future status of the whole network as seen by the local node.

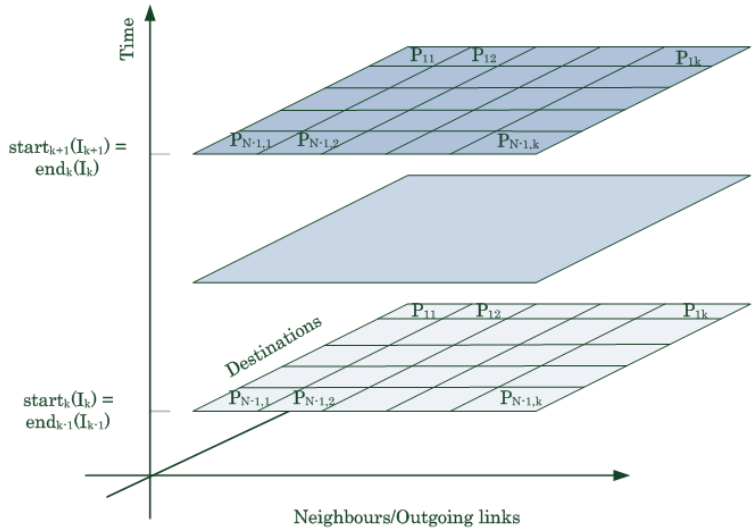


Figure 3.16: Example of the routing table's structure.

$\mathcal{R}_i$ for the case of a node i, with l neighbours and $N-1$ possible destinations. The entries $\mathcal{P}_{nd}, \forall n \in \{1,\ldots,l\}, d \in \{1,\ldots,N-1\}$ are probabilistic values.

Such a matrix $\mathcal{R}_i$ defines the probabilistic routing policy at each node i. For each possible destination $d \in \{1,\ldots,N-1\}$ and each neighbouring node $n = \{1,\ldots,l\}$, during each time slice $\mathcal{I}_k$, $\mathcal{R}_i$ retains a probability $\mathcal{P}_{nd}(I_k)$ expressing the learnt desirability of choosing n as next step to advance closer to the destination d. All the probability values

for going from the current node i to a destination d will have to sum to the value of 1 since an ant can only chose the next node among the neighbours of i:

$$\sum_{n \in \mathcal{N}_i} \mathcal{P}_{dn} = 1, \ \forall \ d \in [1, N-1]. \tag{3.6}$$

This is one specificity of AntNet, shared with ACO algorithms and the hyper-cube framework, that is the sum of all $\mathcal{P}_{dn}$ is normalised to 1.

3.3.3 Solution construction

During the construction of the routing solution from one node to another one each forward ant collects information concerning the problem characteristics and its own performance. The information is gathered in the memory of the forward ant, $\mathcal{S}_{sd}(k)$. Forward ants can be considered investigators that move sensing the delays that will be encountered by cars. From every node s, a forward mobile agent $\mathcal{F}_{sd}$ is launched towards a destination d to discover and determine the minimum cost route and investigate the status of the network. In this way the ants not only determine the shortest paths for the present time but they are training the routing tables in advance for the near future routing. Forward ants experience the same delays as vehicles.

A stack memory $\mathcal{S}_{sd}$ belonging to a forward ant retains every visited node i and the arrival time t_{si} at each one of them evaluated from the launching moment in node s. Accordingly, the stack contains a set of pairs $(i, t_{si}),\ s, i \in V$ in the order of node traversal. The time t_i is the only available feedback to evaluate paths. At each intersection, the forward ant selects the neighbour n choosing randomly among the unvisited neighbours. In case all neighbours have been previously visited, the ant is regarded as inefficient and deleted because a cycle is present in the path selected by the ant. It is considered counterproductive to use the information provided by the ants which have developed cycles to update the routing tables. In [23], Di Caro and Dorigo propose to select the next neighbour n with a probability $\mathcal{P}'_{dn}$ computed as the normalised sum of the probabilistic entry $\mathcal{P}_{dn}$ of the routing table with a heuristic correction factor taking into account the link queue of the edge $e_{in} = (i, n)$. But we prefer to choose randomly the next neighbour and to give more place for the exploration of the graph. This is called trading reactivity to fluctuation with encouraging exploration property of the ant agents.

When the destination is reached, the forward ant $\mathcal{F}_{sd}$ generates an agent called the backward ant $\mathcal{B}_{ds}$ and then dies. The new agent is now in charge of the stack $\mathcal{S}_{sd}$. Using the information accumulated in the stack, the backward ant travels from d to s by popping a node from the stack each time it does a new step. Backward ants need to propagate quickly the information gathered by the forward ants. For this purpose they don't respect the delays encountered by vehicles and use higher priority queues.

3.3.4 Updates of the routing table

Is extremely important how the information coming from the ants is combined in the routing table of each node i. In a network there can be a lot of alternative routes between two points, but from all, only several are short. Pheromone deposit will exert great influence over the path advice and over the distribution of the traffic. So, for a backward ant $\mathcal{B}_{ds}$, arriving at a new node i, all entries of the routing table R_i that correspond to the destination d will be updated. First the backward ant $\mathcal{B}_{ds}$ that arrived in i and which has popped from the stack the pair (i, t_{si}) selects the time interval $t_{si} \in \mathcal{I}_k$ for which the routing table R_i is going to be updated, j is between i and d. This is going to be realized in 4 steps:

1. Get the estimated traveling time on the segment starting in node i and ending in node j (i, j successive nodes) when starting in the interval I_k at node i:

$$t_{ij} = t_{sj} - t_{si}. \tag{3.7}$$

For the description of the computation of t_{ij} see Section 3.4. The other two durations t_{sj}, t_{si} are coming from the stack memory of the ant agent.

2. Update the average travelling time $\mu_j(I_k)$ of node j:

$$\mu_j(I_k) = \mu_j(I_k) + \eta(t_{ij} - \mu_j(I_k)),\ \eta = 0.1. \tag{3.8}$$

The value of η is 0.1. This variable represents what impact, the recent trip times samples, have over the average value $\mu_j(I_k)$. In this way only 50 observations will impact on the average travel time measurement.

3. Compute the reinforcement r to be used to update the routing table. This is a function of the time t_{ij} and its mean value $\mu_j(I_k)$.

$$r = \begin{cases} \frac{t_{ij}}{\alpha\mu_j(I_k)};\ \frac{t_{ij}}{\alpha\mu_j(I_k)} < 1,\ \alpha = 1.1 > 1 \\ \\ r = 1 \text{ otherwhise.} \end{cases} \tag{3.9}$$

The reinforcement parameter r plays a great role in situations when traffic conditions remain static for a long time and then suddenly change. This problem was pointed out also by Schoonderwoerd in [58]. But the solution given at that time was the addition of an exploration probability, or noise, to the random walk of the ants. The solution supposed to ensure that even apparently useless routes are used occasionally, so that at least some information about them is present in the system to give a head start when a route is blocked (e.g. by extreme node congestion or node failure). In that case, it was believed that noise might also encourage the more rapid discovery of a better route which suddenly appears.

However, we believe that having a reinforcement parameter is equivalent or even better than adding random noise. The reinforcement in our case is not random and it is computed as in Equation 3.9. Due to this reinforcement, when the traffic situation changes, and implicitly also the free flow speed and the instantaneous travelling time, the pheromone tables will not be "frozen". This will of course be shown later on.

4. Update the routing table for destination j.

Changes will be performed also on the entries corresponding to $f \in \mathcal{S}_{id}, f \neq d, \forall f \in V$ on the subpaths followed by ant $\mathcal{F}_{sd}$ after visiting the node i. Because all the forward ant decisions depended on the destination node, the subpaths are side effects and are sub-optimal. After all, even though suboptimal paths, these were discovered by the ants at no additional cost.

Considering Figure 3.10 and one forward ant $\mathcal{F}_{AD}$ and the corresponding backward ant $\mathcal{B}_{DA}$ we are going to exemplify how the pheromone matrix is going to be updated. If the destination is $d^{'} = i+1, \ldots, C$ and the current node is $i = C, \ldots, A$, the pheromone matrix $\mathcal{R}_i$ is updated by incrementing the pheromone $\mathcal{P}_{d^{'}f}, f = i+1$ of the last node f the ant $\mathcal{B}_{DA}$ came from (the pheromone suggesting to choose neighbour f when destination is $d^{'}$) and decrementing by normalisation the pheromone values of the other neighbours $\mathcal{P}_{d^{'}j},\ j \in \mathcal{N}_i,\ j \neq f$.

The pheromone mainly depends on the measure of goodness associated with the trip time $t_{id^{'}}(\mathcal{I}_k)$ experienced from i to $d^{'}$ by the forward ant. This time is the only

available feedback to score paths and the best chosen cost since it integrates the number of hops in the network and the delay suffered because of congestion. Nevertheless, the optimal trip time can not be determined because is dependant of the entire network status. A path with a good trip time during a period of congestion might not be so good in case of a fluid traffic. But this is only a disadvantage that ant agents need to overcome. It is a typical problem frequent in the reinforcement learning field.

Update of the routing table for the destination j is done accordingly to the formulas:

$$P'_{jn}(I_k) = P_{jn}(I_k) + (1 - r)(1 - P_{jn}(I_k)) \text{ for the link } i \rightarrow n, \tag{3.10}$$

$$\text{otherwise } P'_{jl}(I_k) = P_{jl}(I_k) - (1 - r)P_{jl}(I_k), \text{ for } l \neq n. \tag{3.11}$$

When the backward ant $\mathcal{B}_{ds}$ arrives in s it is deleted.

3.4 Travel time estimation methodology for a road segment

In thesis thesis we will use the actual route choice to estimate the travel time as the cost of going from one intersection to another one. To apply this travel time estimation, two possible temporal network models were introduced in Section 3.2.2, namely: time-expanded networks an time-aggregated graphs. Only time expanded graphs are going to be used for this project.

Lets suppose that we are given a point A and a point B, both successive intersections in a road network, plus the starting time t. We want to compute the entire duration of the trip between these two points given. Given that the time-expanded network is split in intervals, two different cases arise in the computation process:

1. $t = start_k$, where $\mathcal{I}_k$ is an interval from the sequence $\mathcal{I}_1, \ldots \mathcal{I}_T$ in the time horizon considered.

2. $start_k \leq t \leq end_k$, where $\mathcal{I}_k$ is an interval from the sequence $\mathcal{I}_1, \ldots \mathcal{I}_T$ in the time horizon considered.

   In this last case the starting time t can be anywhere in a time interval, except the start or the end. So the time that needs to be added to the total time is not a whole interval but an unknown fraction of it.

While trying to determine the time to go from start (A) to destination (B), these two cases can combine themselves into different situations on condition that case two can only occur in the end or the beginning of a sequence of cases. For a road segment one driver may need one or two time interval fractions and one or several full time intervals. So there can be repeated sequences of case 1 and 2. To put it more clearly, case 2 can only occur in the end or in the beginning of a sequence of cases. But there is also the situation when only case 2 or only case 1 can form a sequence.

The algorithmic procedure for this methodology is describe in pseudocode in Section 4.3.2.

3.5 Vehicular traffic

Some vehicular traffic has to be generated an incorporated in any test performed on a routing system. In this way, it can be verified if the routing algorithm can be also applied in reality.

3.5.1 Modelling vehicular traffic

Vehicular traffic is one of the essential ingredients in a routing model. Every action performed by one driver affects inevitably the other drivers on the same road. More exactly, what is important for a future route estimation, is how are traffic flows, together with its elements: density or speed, integrated into a routing algorithm. Generated vehicular traffic can also be successfully used to test routing algorithms.

One way to proceed is to generate traffic streams for for a network and then to test how a routing algorithm performs in the conditions of the generated traffic. Another course of action can be based on the assumption that in historical traffic information (time delay or kilometre delay), the traffic flow/density is already included and there is no such need to generate traffic.

The delay that is going to used in the experiments is coming from real measurements. That means that the number of the vehicles on the road at the time of data collection have already influenced the historic traffic delay. So there is no reason to consider that a simulation is not appropriate in the case of this project. For now, only one type of vehicle is going to be generated at a specific time interval to analyze the performance of the new ABC. By type of vehicle we mean that it is actually using only a specific algorithm for routing. In this way the simulation won't require extensive computational power. The disadvantage is that we will not be able to determine the impact that ABC has on the traffic streams.

In the future it would be ideal to assign time-varying vehicles flows (where vehicles are routed trough different algorithms) to a congested time-varying stochastic network. However, the purpose now is to make a simulation that runs fast and that permits a fair evaluation of the ABC algorithm in realistic traffic conditions. Simulating travel demand and real-traffic flow is out of the purpose of this project.

3.5.2 Constraints of network loading

If a simulation is desired, after determining the route choice, the vehicle flows have to be loaded (propagated from source to destination) onto the network. We say vehicle flows because vehicles are launched on the same route at different timestamps. Microscopic, macroscopic or mesoscopic models can be used to fulfil this purpose. The network loading model can be written as a system of equations, representing constraints as: *traffic flow conservation*, *first-in-first-out constraints* and *capacity constraints*.

Flow conservation

Flow conservation means that the traffic flow is neither generated, nor lost in the network. But this happens without taking into account the source and destination intersections. For a graph $G = (V, E, L)$ where every edge $e_{ij} = (i, j) \in E$ has a non-negative capacity of vehicles and $F\colon V \times V \to R$ represents the flow, then the conservation equation can be written:

$$\sum_{a \in V} F(i, j) = 0, \text{ unless } i \text{ is not the source of the flow and } j \text{ the destination.} \quad (3.12)$$

For every time instant t, the flow into node n at time t minus the flow that has destination n at same time instant should be equal to the flow out of node n at time t minus the flow that originates from node n at time t, Figure 3.17. This is an extension from the flow conservation constraints of static traffic assignment. This constraint makes sure that flow entering a certain link exits the link again after the link travel time elapses. When a vehicle enters a link it faces the traffic on that link at that specific time.

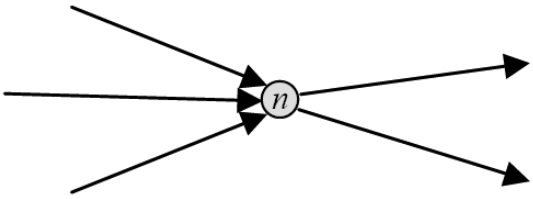


Figure 3.17: Conservation of flow in a network.

First-in-First-out (FIFO) link constraint

This constraint says that a vehicle which enters a link first should exit the link first. Critiques may say that overtaking is plausible and then the constraint fails. But very often in assignment models, the flow of vehicles is assumed to be homogenous. If vehicles enter later but exit earlier and FIFO is violated, then this is not consistent with the user-optimal idea.

Capacity constraint

Each link has a physical maximum inflow and outflow (vehicles per hour) depending on the number of lanes and the link type. The maximum outflow is referred to as the capacity of a link. The capacity may vary due to traffic conditions and traffic management. The sum of outflow capacity of a link is equal to the sum of the inflow capacity of the consecutive links.

Capacity limits the flow over links such that congestion may occur. Moreover, it is clear that the capacity influences link's travel time.

We have to mention that the current implementation does not work with capacity of the links, from the obvious reasons that:

- we do not know how many vehicles were present on the roads, when the historical traffic measurements were gathered,
- and because we don't assign the capacity or the demand for the spatial network.

Chapter 4

Implementation details

This chapter is devoted to the presentation of the implementation details concerning the new AntNet. All the specifications summarize the development tool's main characteristics, the system's architectural design, the important pseudo code parts and the GUI description of the application.

4.1 Development tool: Quintiq

For implementing the model we had to find an environment that could fulfill several requirements:

- it should contain an object-oriented programming language,
- provide facilities for developing a graphical user interface that includes a map display,
- have access to databases.

The main candidates were quite a few, including frameworks like Sun's Java and Microsoft's Framework.NET, Quintiq Environment etc.

We chose the most complete and new tool for the application that we want to implement, yet the most unknown to the general public. Our routing system will be developed in *Quintiq Environment, version 4.2.5* (shortly said Quintiq) and *Quill* language, using the maps and coordinates provided by the *PTV eMapServer* and *eLocateServer*.

4.1.1 Motivation

We have chosen this development environment according to the restrictions under which this application needs to be developed and due to the components that are provided for the development. Even though is not a free distributed environment we were encouraged to choose it due to the integrated *GIS components* available in Quintiq Environment, due to the *client-server* architecture and because it permits the designer to create structured graphical framework without worrying about low-level details. A plus is added by the availability of a comprehensive map of the Netherlands or of other regions.

Other, crucial advantages of Quintiq tool are: the system can be adapted to suit individual needs in a user-friendly way and is easy to learn and utilise. Another point that encouraged us in using Quintiq is that this company has been involved for many years in solving Traveling Salesman Problem (TSP) and other logistics problems. Moreover, the Man and Machine Interaction Group of TU Delft has received an academic license and permission to use this environment for developing a routing application.

Based on the introductory manual of Quintiq [55], we can make the following observations about Quintiq environment:

- The Quintiq business model is object-oriented. Inheritance is therefore very important, and especially inheritance from standard (coded) types is a powerful mechanism.
- Easy to develop the Graphical User Interface (GUI) and auto complete help in developing complex application.
- Quintiq has an interface with the geographical information system (GIS) supplied by PTV. Quintiq can access the GIS and extract information about the locations of places, the distance between them, and the duration of the journey between them, and display this information on a graphical map within the Quintiq application.
- Debugging mode via an Error Warning window, Server Output and log files. Moreover, Quintiq has a System Monitor incorporated in its architecture.
- Quintiq Environment enables the realisation of decentralised systems, where the computation is done locally, for individual clients.
- It is a suitable solution for routing algorithms because it offers besides an own modality of routing between two coordinates, the possibility to develop new algorithms. We need to mention that we will create our own dynamic routing algorithm.

A negative observation that needs to be made is that Quintiq Environment lack multithreading support.

4.1.2 Quintiq environment

Quintiq environment (see Figure 4.1) has a three-tier client/server architecture that consists of the following (physical) modules:

- Server,
- ODBC Integrator (dbodbc),
- Messaging Integrator,
- Dispatcher,
- Windows Client,
- Thin Client Engine,
- Thin Client,
- Web Interface.

In the lines that follow we will only discuss the modules that present interest to the current project: the Quintiq Server, the ODBC Integrator and the Client along with the Client Engine.

The *Server* is the application that performs services for the connected clients as a part of the server-client architecture. In this case, the server runs the business model. As such it contains the business model, the business rules and has access to all the data, in connected DataBases or Knowledge Bases (KB). All the other modules may view this (Meta) data, but only the server itself is allowed to modify the structure, logic or the data in order to ensure consistency.

The *ODBC Integrator* module provides connectivity facilities to any ODBC level 2 compliant databases. One installation of Quintiq may contain multiple ODBC integrators to provide connectivity to multiple databases. Each integrator communicates through its own TCP/IP port. This port is selected using the Quintiq Configuration Utility. For

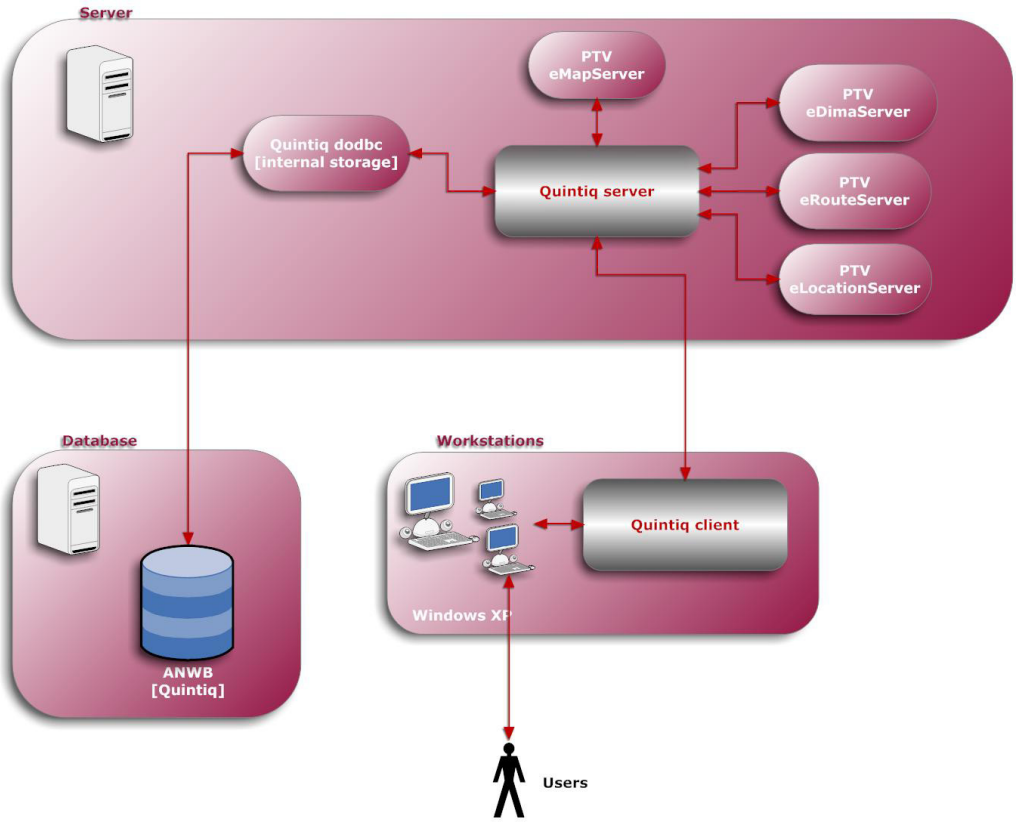


Figure 4.1: Architecture of Qunitiq Environment, version 4.2.5.

each copy of an integrator, a separate tab containing the parameters for that integrator is added to the configuration utility automatically.

A *Client* provides a way to interact with the model in a Windows environment. Clients can be connected either directly to the server or to a dispatcher. Connecting a client to a dispatcher allows the data being transmitted to the client, to be filtered.

The *Quintiq Client Startup Dialog* enables the user, as well as the programmer, to start a *Business Logic Editor*, a *Designer* or a *Quintiq Application*.

From of all of these three workspaces the most important is definitely the Business Logic Editor that enables the programmer to create the model and the business logic of a customer's application. The *main editor window* consists of a *workspace* and a *model overview* that shows the model structure as well as the types and the relations between them. Even if at the heart of a Quintiq application is the business logic model, the interaction between the user of the application user and the business logic model is done through a graphical user interface (GUI), consisting of basic dialogs, menus, toolbars and so on. The Quintiq GUI Designer (or simply the Designer for short) is the tool that we used to create a GUI.

4.1.3 Quintiq and databases

Almost every dynamic application eventually boils down to accessing, manipulating, and presenting information. That is the reason why real-time systems often require a connection with a database. Furthermore, this is vital because almost every application calls for some kind of interaction with the data stored in the database.

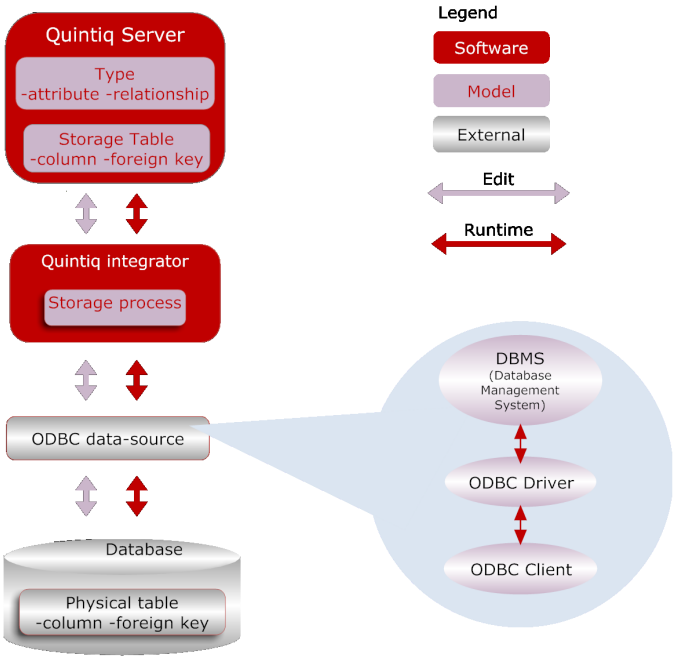


Figure 4.2: Quintiq Environment's interaction with databases.

The scheme in Figure 4.2 gives an overview of the layers necessary to communicate with a database in Quintiq environment.

Since the internal layers of Quintiq have already been discussed in the next lines we will only present the external part of the relation with a database. When working with the dynamical data the real connection with the database is one of the most important things. The power of the application lies in the ability to extract the required information. A common used tool for creating this connection is the ODBC or Open DataBase Connectivity. This is a standard database access method developed by the SQL Access. ODBC is a core component of Microsoft Windows Open Service Architecture and makes it possible to access any data from any application, regardless which database management system (DBMS) is handling the data.

Three main components should be mentioned when discussing ODBC: ODBC Client (for example: Microsoft Access), ODBC Driver and DBMS Server (which can be either SQL Server, Microsoft Access, Oracle or any DBMS for which an ODBC driver exists). Further on we will explain how the ODBC works. The client uses a language or vocabulary of commands to request data from, or to send data to, the server DBMS. However, the DBMS doesn't understand the ODBC client request until the command passes through the ODBC Driver for that specific DBMS. The ODBC driver also translates the command into a format that the ODBC Server can understand. Consequently, the driver purpose is to translate the application's data queries into commands that the DMBS understands.

4.1.4 Quintiq and PTV

Information is often much easier to understand when it is displayed on a map. Quintiq therefore has an interface with the geographical information system (GIS) supplied by PTV. Quintiq can access the GIS and extracts information about the locations of places, the distance between them, and the duration of the journey between them, and display this information on a graphical map display within the Quintiq application.

There are four servers in the PTV system:

1. **a locate server** (*eLocate server*), which assigns a latitude-longitude coordinate to an address (a process called Geo-coding). Once a latitude-longitude coordinate has been assigned, the address can be displayed on a map or used in a spatial search.
2. **a distance matrix server** (*eDima server*), which provides a distance matrix from which the distances between locations and the journey duration between them can be retrieved.
3. **a route server** (*eRoute server*), which maintains the details of routes and the stops within the routes.
4. **a map server** (*eMap server*), which draws maps and shows locations on the maps in Quintiq.

The *DIMA Server*, the *route server* and the *locate server* are accessed by the Quintiq Server to retrieve the graphical information used in Quintiq models.

The *map server* and *route server* are accessed in the Quintiq Client to show data graphically. These two servers are the graphical part of the GIS system used to draw a map and show regions, routes and locations on the map.

The *PTV eDima server* is normally used for retrieving the route information between two locations. However, the eDima server is meant to be used to retrieve a complete matrix of route information for a set of locations.

The *eRoute server* also has specific detailed route information that is not available in the eDima server (such as ferry crossings and border crossings) that can be made available to the Quintiq application.

4.2 System architectural design

A prototype system architecture was created to support the routing algorithm. The system will serve *en-route* routing planning to travelers. Accurate advice for the next best intersection to follow will be provided to drivers. But the system that we are implementing it is a simulation that assumes more vehicles at the same time.

Among the first task undertaken when implementation of a system is concerned, is a clear decomposition of the application into appropriate software components. Object-oriented analysis and design methods are becoming the most widely used methods for computer system design. Moreover, the application (*Quintiq Environment*) used for the implementation of the proposed ABC routing algorithm obliges us to construct an object-oriented solution. The heart of object-oriented problem solving is the construction of a model. The model abstracts the essential details of the underlying problem from its usually complicated real world. We will use UML for analysing our software design choices.

In Chapter 3 we described the model design which will be used for routing drivers between source and destination. Altogether, a motivation to the choices is given there. The decisions made for building the software system are based on the knowledge we have gathered so far.

4.2.1 Chosen architecture

Figure 4.3 displays, from a system engineering perspective, the components of the routing system as we have implemented it.

The idea is to get a reliable comparison between our algorithm and similar ones. We are more motivated to produce simulation results than to discover how ABC can be implemented as a client-server architecture, with several clients that compute their own route.

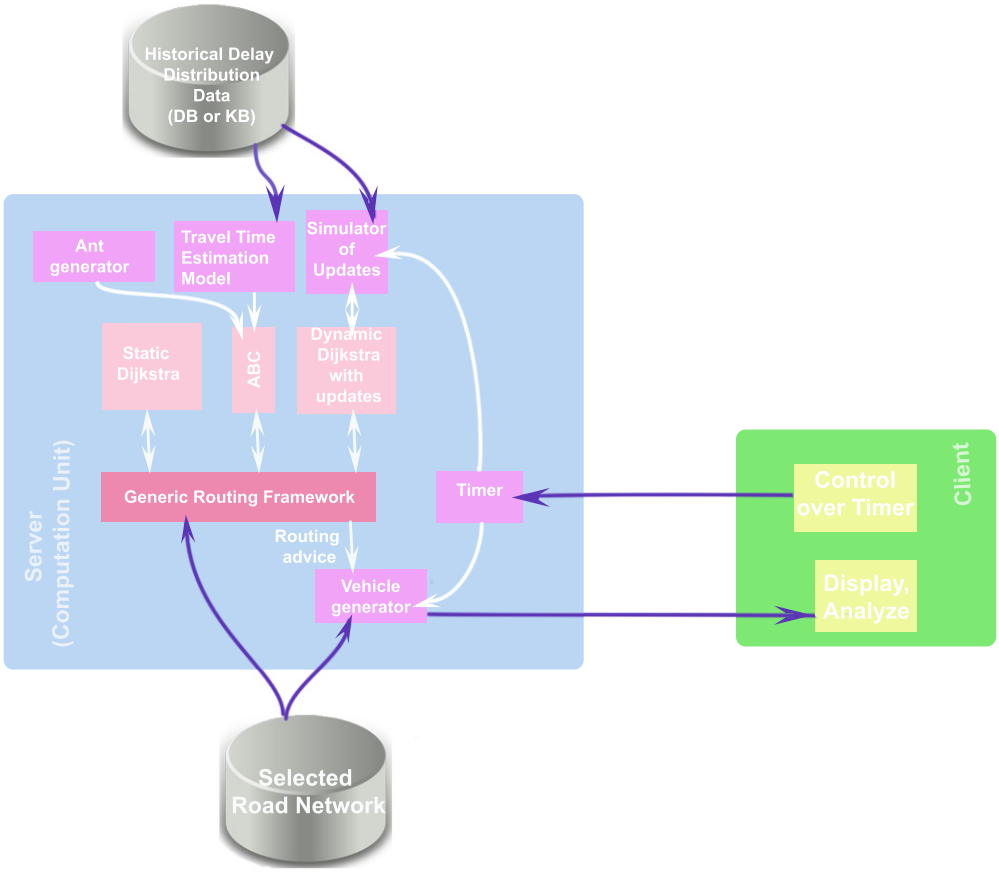


Figure 4.3: Architecture of the routing system as it was implemented in Quintiq.

Of course, this would be the ideal case. Because we want to compare ABC with Static Dijkstra (SD) and Dynamic Dijkstra (DD) with updates, a question that arises is: should these algorithms also be distributed so that they are comparable? Nevertheless, ABC is based on local information, which makes it extremely suitable to be implemented as a distributed architecture. What is more, Quintiq Environment encourages client-server architectures. But then if we want a distributed architecture, navigation devices should have sufficient computation power to manage the route computation for large networks. These devices exist, but for this project we don't have access to such technologies.

Even though Figure 4.3 is quite self-explanatory we give also a textual explanation here. In this current architecture, events are generated because of a *timer* that simulates the flow of time. An event generated because of this timer is the creation of vehicles. The generated vehicles receive their routing advice from a *Generic Routing Framework*. This framework has three integrated routing algorithms:

- *ABC*, which functions on the basis of an ant generator and disposes of a new *Travel Time Estimation Model* (see Section 3.4 for that),
- *Static Dijkstra* that was detailed in Section 2.2,
- and *Dynamic Dijkstra* with updates (see Section 5.1.3 for details).

ABC is based on a *Travel Time Estimation Model*, which gets its data from a *Historical Delay Distribution Data*. The last one can take the form of a *Knowledge Base*, called

also KB (that gathers several tables which are read in the application), or the form of a *Database* (DB). We prefer to use the first one when the simulation is started, otherwise the DB is just fine. Through this decision we eliminate the extra time necessary to read/write data in the DB. The reading and writing procedures from and into a DB are not highly optimised operations in the implementation tool that we used: Quintiq Environment.

Because it is a simulation and we want to analyse all the results together, a single Client displays the following:

- the routes and their average travel times at different times of the day,
- plots the variable free flow speeds in the network,
- displays the pheromone tables at different moments.

The client offers also control of the timer and integrates map displays of the network.

4.2.2 UML diagrams

The entire software that was realized for testing the modifications brought to AntNet can be described via the unified modeling language (UML) approach. According to this approach we can focus on detailing the inner workings of the system. For the sake of clarity, in this section we discuss the use case diagram, the class diagram and the sequence diagram.

Use Case Diagram

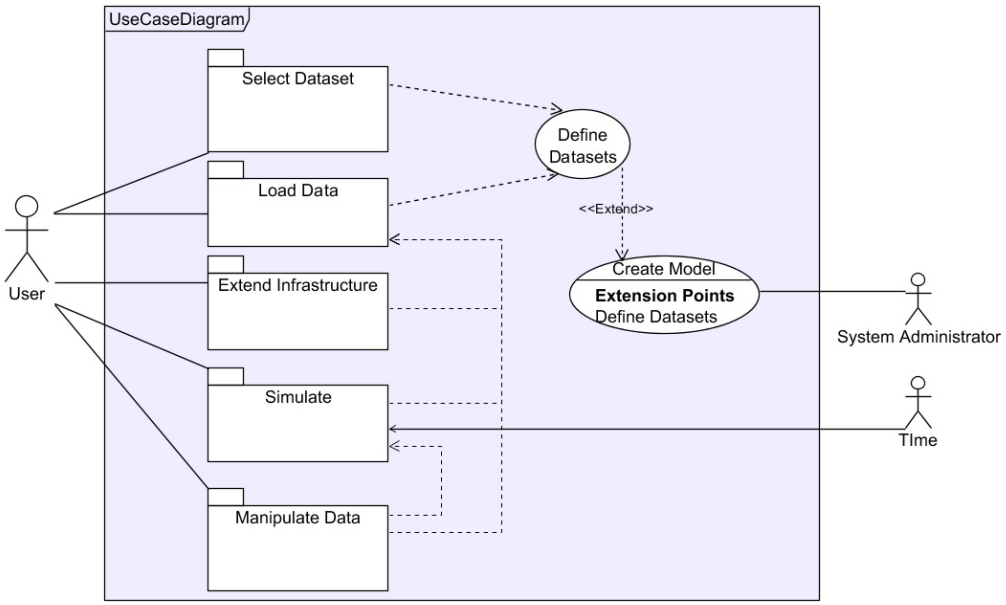


Figure 4.4: Use case diagram illustrating main functionalities of our routing system.

To simplify the diagram we grouped the use cases into packages. Packages' description can be found in Figures 4.5 - 4.7. The interrupted lines indicated dependency relationships.

According to the *Rational Software* practices of IBM, the main purpose of the use-case diagram is to provide help in visualizing the functional requirements of a system, including the relationship of "actors" to essential processes, as well as the relationships among different use cases.

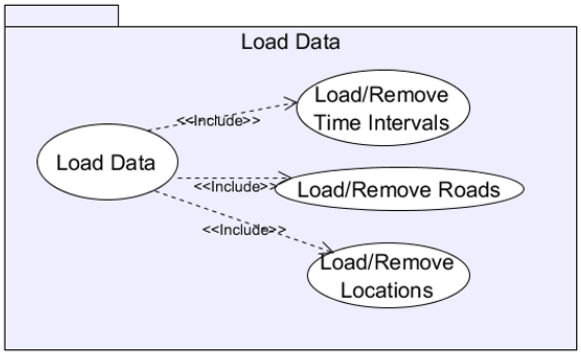


Figure 4.5: Use case package for loading data.

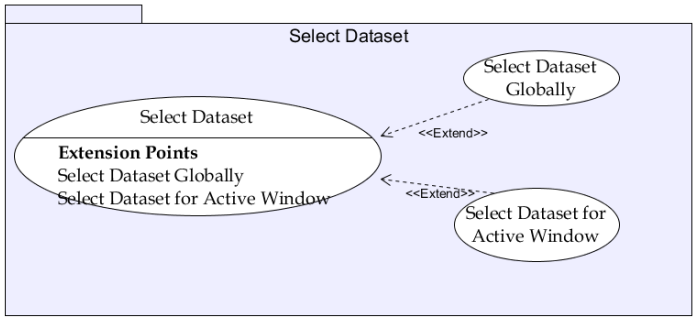


Figure 4.6: Package representing the dataset selection use cases.

A use-case diagram is typically used to communicate the high-level functions of the system and the system's scope. Use case are useful to model the required behaviour of a complete system or portions of a system. Next, we do not assume that our use-case diagram is entirely complete but we try to keep the focus as simple as possible on the essential functionality provided by the system.

By looking at our use-case diagram in Figure 4.4, you can easily tell the the basic functionality that our system provides to the user and the system boundaries. This use-case diagram was created from the user's point of view. Because experts often suggest that a use case diagram needs to have maximum 7 use cases we grouped them into packages. These packages are depicted in Figures 4.5 - 4.7 and the techniques has been inspired from [1]. Packages enable to organise the model elements (such as use cases) into groups so that large diagrams become smaller and more readable.

There are three actors that interact with the system, namely: a user who interacts with the routing system, a system administrator whose main purpose is to create the model and an actor called "Time" that initiates scheduled events. Users select the appropriate dataset after which they load the data as in Figure 4.5 (time intervals that contain the cost of the network, the roads and the locations of specific intersections). In principle, the dataset selection and loading depends on the model created by the system administrator. There can be specific datasets defined. These datasets contain instances of the types/classes defined by the system administrator. The primary use for datasets is to allow application users to use many scenarios. They can load and unload as many datasets (scenarios) as they like, each reflecting a different situation. A user can choose to select a global dataset or to choose a dataset for the active window, (Figure 4.6).

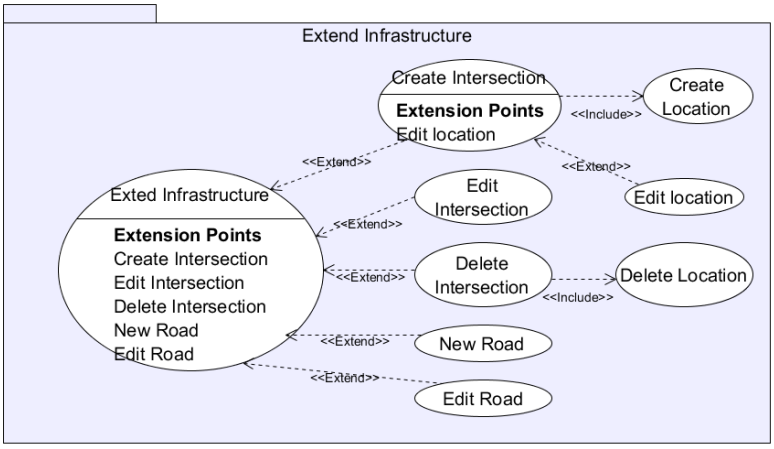


Figure 4.7: Extend infrastructure related use cases.

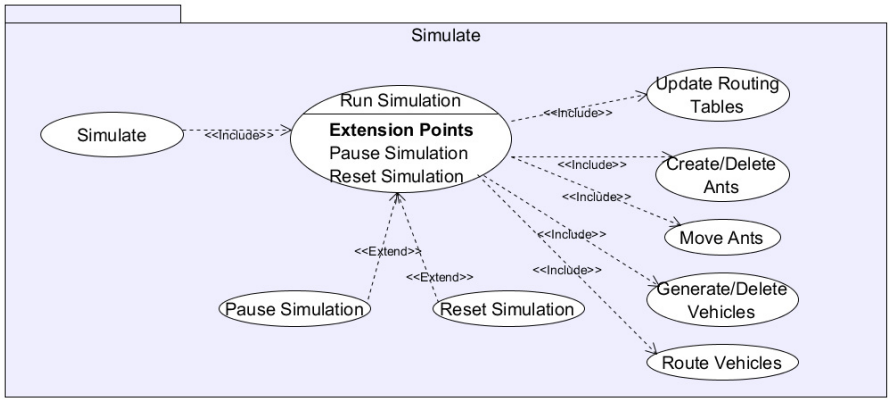


Figure 4.8: Representation of the simulate use cases.

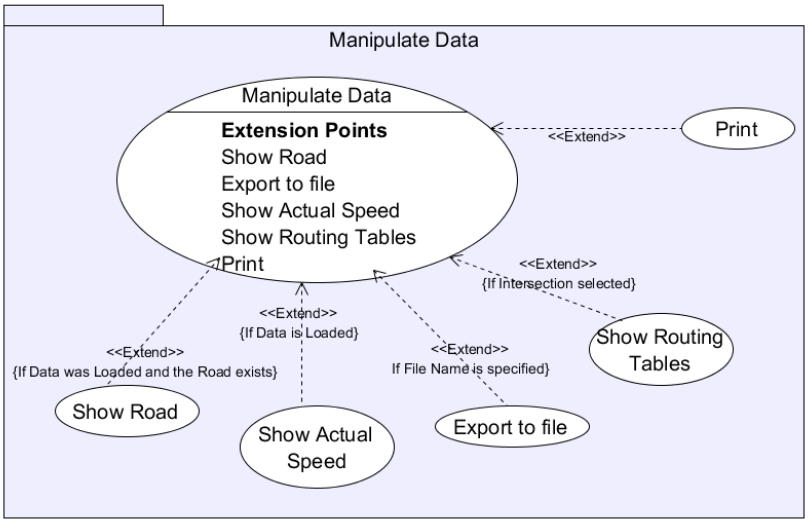


Figure 4.9: Manipulate data package of use cases.

Although the dataset selection was not implemented by ourselves and is a part of the Quintiq functionality, this is quite important for the working of the application. Moreover, we consider that code reusing is always valuable. Our routing system is connected with an external datasource (database) and can load or modify data belonging to it.

For the relationship with the datasource to be meaningful, there has to be a relation between the database storage and the model itself (class model). Therefore, we have created the classes in the model with which the external data is associated. A class was created for each external storage table and then the relations between the classes were defined. The external database structure mirrors the class model. The mapping of the external data onto the class model itself is part of the external data configuration. The tables in the database and their structure are controlled by Quintiq and so there is very little to configure. During a run, the root of the storage is a dataset, and in this case it is known as the modeled dataset (MDS). As mentioned previously it contains instances of objects. A dataset can be loaded into or unloaded from the database. When creating dataset the storage mode must be specified. A dataset can be created either with full storage, when the complete structure of the dataset is stored in the database, or memory only, when we don't want the dataset to be stored in the database at all. In our case, the dataset that will be loaded in the database is owned by Company.

Coming back to the use case diagram, users can extend the infrastructure, or the graph that represents the traffic network by adding intersections, modifying or deleting them and by creating and editing roads (Figure 4.7). Before creating an intersection, several locations (exit points from a road) must be created. The user is also responsible for starting/pausing and restarting the simulation (Figure 4.8). This particular simulation includes updating routing tables, creating ants or deleting them if they have reached their destination, moving the ants from node to node in search for a new path. Nevertheless, we don't forget about generating and routing vehicles. Vehicles that have ended their mission are deleted. Simulation depends on the existing infrastructure and the data loaded.

The user has the possibility of visualising the roads created on the map, visualising a graph of the actual speed and analysing the routing tables of each intersection. Moreover, exporting to file and printing are also part of the existing options for manipulating the existing data or the data resulted from the simulation.

Class Diagram

A class diagram shows the classes of the Object Oriented system, their inter-relationships and the operations and attributes of the classes. Its main role is to represent the conceptual model which depicts our understanding of the problem space for the routing system. The class model of this application is a complex structure that can be tedious to understand if all the elements (attributes and operators) of each class are depicted. It is in our benefit to make a simplification, the UML diagram in Figure 4.10 illustrates the stylised class model. The full class diagram is to be found in Appendix A. In general, a class diagram describes the structure of any system and gives an overview of the target system. The domain is the actual world from which the problem comes. Specifically, classes are abstractions of the attributes, the behaviour of a set of objects and any additional information about them. Class diagrams are static because they display what interacts but not what happens when they do interact.

The Quintiq business model is object-oriented. Inheritance is therefore very important, and especially inheritance from standard (coded) types is a powerful mechanism. Quintiq allows only single inheritance. This means that an inheritance hierarchy is built up in which each type has (at most) one parent, also called its *base type*. In Quill (the language of Quintiq Environment) all the information about how the world model looks is described by types. A type is a description of a specific object. Two different kinds are available

in Quill, namely *object types* and *user-interface types.* Both contain a lot of the same functionality, however, user-interface types work in a slightly different way. We are going to refer first to the object types for an UML description of the application (Figure 4.10). The user-interface and its types will be treated later on.

Object types (also referred to as Server types since these types are defined by the Quintiq server or are part of the business model that is maintained by a server) are normally either coded types or modelled types defined in the business logic editor. Each object type is made up of a number of elements, namely: attributes, relations, methods, functions and other elements that we do not use. The types represented in the class diagram (Figure 4.10) are object types. In addition to what has been mentioned earlier, an object type also supports three native "methods", namely *OnCreate*, *OnInitialize* and *OnDelete*. Each of these three statement blocks contain procedural logic describing how an object is going to be initialised the first time is created or in other cases destroyed.

In our case the main object type or the central class is **Company** which models the simulation of the routing process in a realistic environment and realistic travel times. Also, there are 4 base types that we use, namely: Object, GISLocation, Sequence and SequenceElement. Sequence assembles a set of ordered SequenceElements, while GISLocation is used to display locations on a map or to compute distances between locations using GIS system (geocoding). For the Company class, the base type is Object. In Figure 4.10, the classes are grouped by colour, according to which base type class they inherit from. Moreover, from the three native methods available for a class, we only use *OnCreate*, which in the case of objects read from storage (for example a DataBase) is not executed. Among all the mentioned classes, only 3 have as native method OnCreate, namely: Company, Intersection and Road.

The class Company is in a composition relationship with several other classes: Ant, Intersection, Parameters, Route, Vehicle and Location. The second class in order of importance and number of relationships after Company is Intersection.

Sequence Diagram

Apart from the class diagram, a sequence diagram was also developed to capture the flow of logic within the system in a visual manner. Generally speaking, a sequence diagram depicts the sequence of actions (interactions) that occur in a system between classifiers in the sequential order that those interactions occur. A sequence diagram can be created either at the system level or at the object level.

In this case a sequence diagram that reflects, in a simplified manner, the interactions found at the server level was created. The reader may find it in Figure 4.11.

Table 4.1: Groups of classes that belong to the routing system and their purpose.

Classes	**Purpose**
Location, Intersection, Road	These 3 classes describe how the road infrastructure was implemented. Physically, an intersection (represented by **Intersection** class) is constituted of several highway exit points, that we represent by **Locations**. Locations can be the start or the end of a road. In our logical structure, each Intersection has associated several Nodes. This is to say that each Intersection can be a node in the memory of a couple of Ants. Intersections have a name and some geographical coordinates that will help us in representing them on a map. The edge that connects 2 locations belonging to separate intersections is called a **Road**. This explains the use relationship between Road and Location. Roads have as attributes maximum speed, direction, length and capacity.
RoadCalendar, RoadTimeInterval	Moreover, a Road owns a **RoadCalendar**, which in turn contains **RoadTimeIntervals**. The last one has as attributes a start time and an end time or it can state that the road is closed. In RoadTimeInterval we encapsulate the delay time experienced when traffic congestion occurs.
Ant, Memory, Node	The **Ant** class takes care of the details concerning a stigmeric agent: its type (Forward Ant or Backward Ant), the current Node, the source and the destination Intersection. The Ant class implements the move operation from Node to Node as well as adding the previous Nodes into Memory. An Ant can also detect a cycle in its path. There is a strong life cycle associated between Ant-Memory and between Memory-Node. Memory's lifetime depends on the Ant and Node's lifetime depends on the Memory of the Ant. **Memory** inherits Sequence, which is composed of an ordered set of SequenceElements. For Memory, the SequenceElements are Nodes characterized by the attribute ArrivalTime.
PheromoneTableSequence, PheromoneTable, PheromonePerDestination, PheromoneValue	Ants act on the basis of pheromone. For retaining the pheromone values between different intersections starting at different time layers we needed a 3-dimensional table. Because of this and because Quintiq doesn't recognise such data structure we designed these 4 classes. At the same time an Intersection owns a couple of **PheromoneValues** and a **PheromoneTableSequence**. PheromoneTableSequences own PheromoneTables for every time layer, that further own **PheromonePerDestination** sequences with PheromoneValues. Here the update of the probabilities takes place every time a Forward ant has ended its mission. The association between PheromonePerDestination and Intersection has just an implementation purpose.
Vehicle, VehicleSequence, CarOnLink	Class **Vehicle** is an abstraction of a real car on the road. And like real cars can be guided by different routing algorithms that we will detail further on. These vehicles go from source to destination by moving forward trough several Roads. Any Road owns a **VehicleSequence** which is constituted of many **CarOnLink**. This last class was necessary because elements in a Sequence can not be double owned. Each CarOnLink has its own position and is in an association relationship with a Vehicle.
Route, Stop	The results of the routing process, namely **Stops** with their arrival time form together a **Route**.

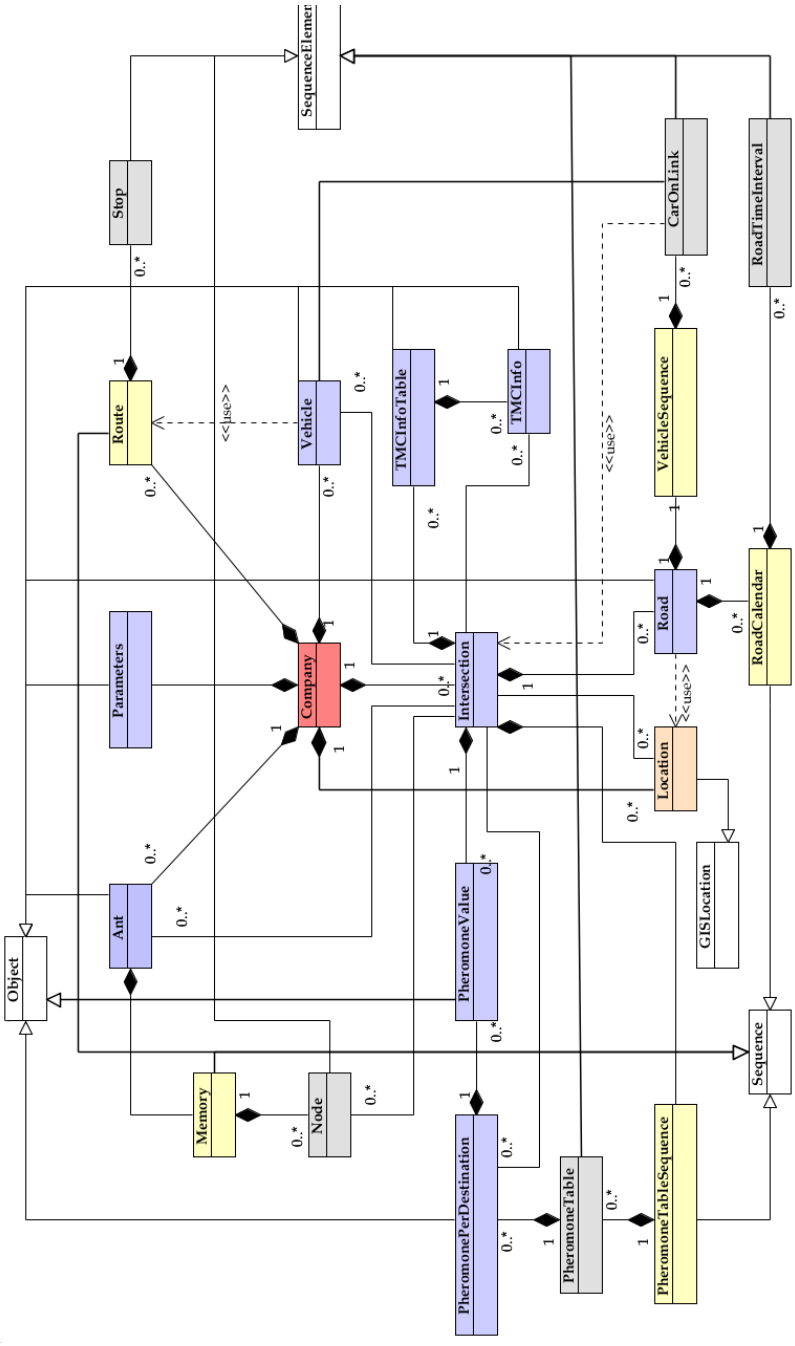


Figure 4.10: Simplified Class Diagram.
The central class is Company. The blue coloured classes inherit from Object, the yellow classes inherit from Sequence, the grey coloured classes inherit from SequenceElement and the peach coloured class inherits from GISLocation.

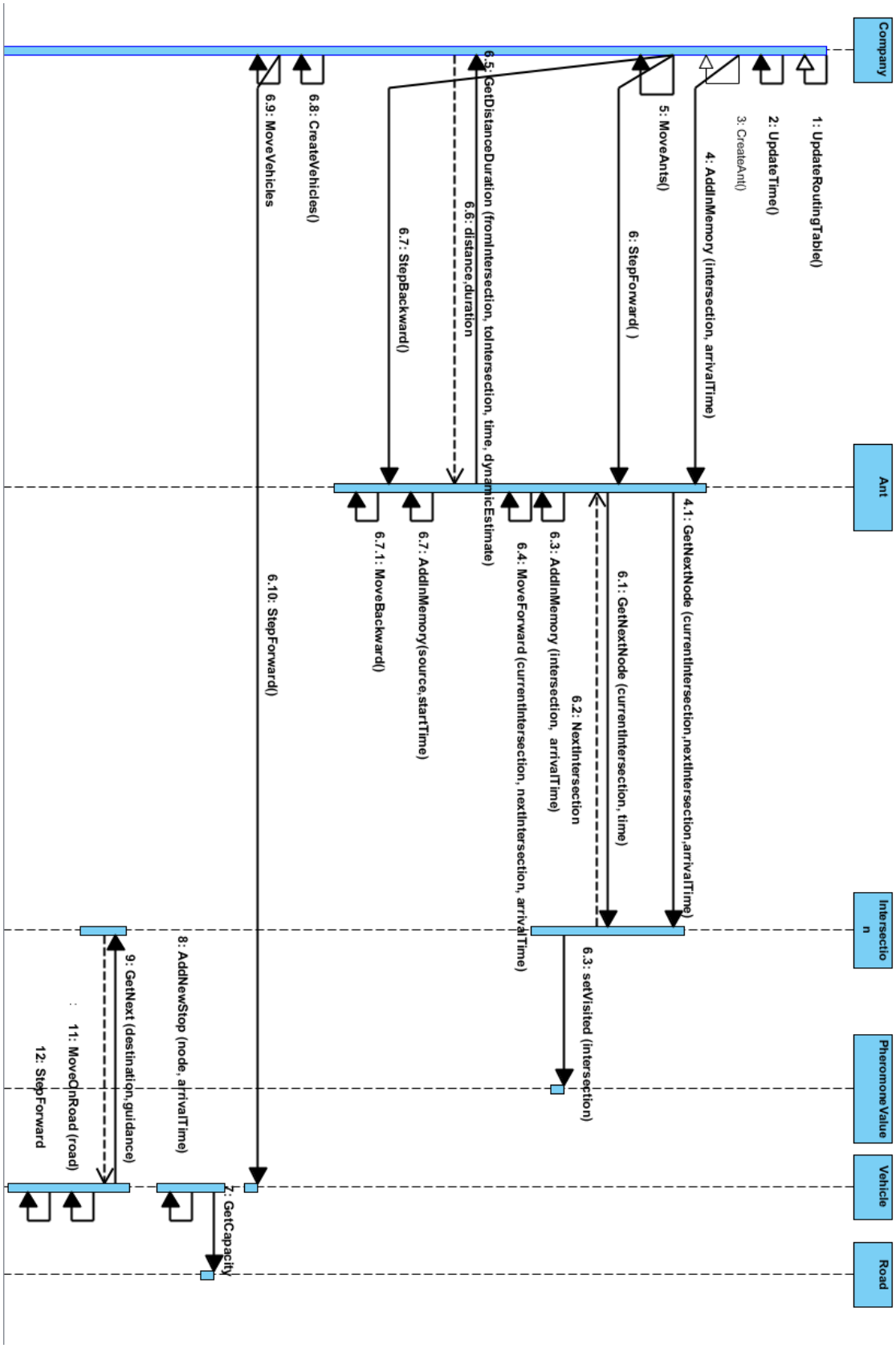


Figure 4.11: Simplified Sequence Diagram for interactions at the server level.

4.3 Algorithm pseudo-code description

4.3.1 ABC algorithm

A pseudo-code description of the actions performed by the set of forward and backward ants, that are active in the network, is reported in Algorithm 5. This is the main structure of the algorithm.

Algorithm 5 Pseudo-code description of ABC algorithm.

The actions of the whole set of forward and backward ants active in the network are executed in a totally distributed way.
At certain time steps, forward agents F_{sd} are generated for a specified time horizon, comprised of k time intervals. For every forward ant agent received in a node, the ant is removed if a cycle is detected. If the destination d is reached, a corresponding backward ant B_{ds} is created and the forward ant F_{sd} is killed. Otherwise if the destination is not reached, $GetNext$ selects the next intersection n for the forward ant and estimates the traveling time t_{sn} necessary to reach it by a vehicle (see Algorithm 6). The new information is added to the stack and the forward ant F_{sd} moves another step closer to the destination d.
Next, for all the previously created backward ants, the update of specific probabilities for the current node i of the backward ant and the destination d of the forward ant is performed. Then, the backward ant moves closer to the source. When it reaches the source s the backward ant is also removed.

$ABC()$
  {i - current node, d - destination node, s - source node}
  {n - successor node of i, p - predecessor node of i}
  for all $elements \in V$ **do**
    if time to generate an agent at node s **then**
      for all now and next k time intervals **do**
        Create $F_{sd}(start_k)$
      end for
    end if
    for all forward ants $F_{sd}(t_{si})$ received at node i **do**
      if cycle detected **then**
        Remove $F_{sd}(t_{si})$
      else
        if $i = d$ {destination reached} **then**
          Create $B_{ds}(t_{sd})$
          $p \leftarrow GetPrev(F_{sd}(t_{sd}))$ {select previous node in the stack}
          Move $B_{ds}(t_{sd})$ to p
          Remove $F_{sd}(t_{sd})$
        else
          $n \leftarrow GetNext(F_{sd}(t_{si}))$
          $t_{sn} \leftarrow t_{si} + GetTravelTime(i, n, t_{si})$ {compute the traveling time}
          $F_{sd} \leftarrow (n, t_{sn})$ {add the new information on the stack}
          Move $F_{sd}(t_{sn})$ to n
        end if
      end if
    end for
    for all backward ants B_{ds} received at node i **do**
      $UpdateProbabilities(i, i \leftarrow d)$ {update the node information}
      if $s \neq i$ {source not reached} **then**
        $n \leftarrow GetNext(B_{ds}(t_{si}))$ {select next node to go to}
        Move $B_{ds}(t_{sn})$ to n
      else
        Remove $B_{ds}(t_{si})$
      end if
    end for
  end for

4.3.2 Dynamic travel time estimation

Next we present the algorithm used to compute the travelling time $t_{ij}(I_k)$ between i and j (i and j successive intersections) when starting in interval I_k. Algorithm 6 is integrated into the ABC pseudo-code.

Algorithm 6 Pseudo-code of procedure *GetTravelTime*.

This procedure implements the travel time estimation methodology for two successive nodes i and j as it was proposed in Section 3.4.

```
GetTravelTime(i, j, t_i)
  {i - current node, j - successor node of i}
  {t_i - departure time from node i}
  L ← d(i, j)
  t ← t_i
  k ← 0 {start the iteration with the first time interval of that road}
  while k ≤ n and L > 0 do
    if t ∈ I_k then
      {check if the remained distance can be traversed in this time interval}
      if L ≤ v_ij(I_k)(end_k − t) then
        t ← t + L / v_ij(I_k)
        L ← 0
      else
        L ← L − v_ij(I_k)(end_k − t)
        t ← end_k
      end if
    end if
    k ← k + 1
  end while
  return t − t_i
```

4.4 Implementation issues

4.4.1 Vehicle generation

As we have mentioned before, the current implementation is a simulation based on historical measurements. The flow of time is simulated for the selected traffic delay information. When selecting this information an assumption was made, namely that travel time delays distribution of every road in the network which form the historical information imply a certain number of cars on the road.

Due to the mentioned assumption, only one type of vehicle is going to be generated for every time step of the simulation. The type is a vehicle who uses a specific routing algorithm. In this case there are 4 types of vehicles, vehicles that use either Static Dijkstra, Dynamic Dijkstra with updates every 10 minutes, or every 30 minutes and vehicles that use the new ABC. One can also consider that the proportion of vehicles is 25% of each type. For details see Chapter 5.1.3.

4.4.2 Modelling the future

Even though in the beginning of this thesis we expressed the desire to offer a routing advice (fastest route) that will avoid future traffic congestion, the solution to this problem is not so straightforward.

When congestion forms in a road network, an excessive density of cars on the roads leads to an average travelling speed that is below the maximum travel speed of those

particular roads. In order to avoid traffic congestion, an intuitive approach would be to recommend detours based on the estimated future densities.

In [70], the interdependency between average vehicle speed, a street's vehicle density and the traffic quality (in terms of congestion) was approximated using a model based on empirical studies. The result of the approximation is the density-speed-curve, for 6 and 4 lane freeway and for highways. However, this is not possible in this implemented simulation which is only based on two days of historical data. Needless to say, two days of historical data are two few to attempt predicting the car density on the roads. Moreover, a specific course of action was not established for that. So we have to launch ants not based on predicted vehicle densities, but so that the pheromone tables are fully trained to anticipate congestion.

Three types of exploratory ant agents are created across the network:

- from each intersection to any other intersection,
- for each source and destination of the vehicles,
- for each vehicle in the network that uses ABC.

Even though the pheromone tables are initialised in the beginning based on the distances of each road segment in the network, some additional precautions were taken to avoid a long adaptation time of the pheromone tables in the first runs of the algorithm. The first two types of ants will train the pheromone tables for this purpose when no vehicles are present in the network.

Ants are launched at each simulation step. Because an ant can reach the end of the road in several time intervals (see Section 3.4), the ant can reach other time layers in the time-expanded network in Figure 3.8.

4.4.3 Parameters

As long as the user keeps the simulation running, the time is updated by the server, but also the routing tables are updated by the ant agents running in the network, as it was described in Section 3.3.4. For this simulation to work, certain parameters are required.

The parameters of the simulation can be found in Table 4.2. First 4 parameters concern the ants and their exploration process, while the next three represent the interval between considered traffic updates. The last parameter is the timestep of the simulation. These parameters can be changed if one desires to. Moreover, these parameters apply to all the tests that are going to be done in the next chapter.

Table 4.2: Parameters of the implemented simulation.

Parameter	**Value**
Duration between Pheromone Tables	0:05:00
Pheromone Table horizon	2:00:00
η	0.1
α	1.1
Duration between Static Dijkstra updates	1 day
Duration between Dynamic Dijkstra 10 min	0:10:00
Duration between Dynamic Dijkstra 30 min	0:30:00
Timestep of the simulation	0:01:00

4.5 System interface description

Usually, interfaces are quite valuable for a system because they clarify the output of the collaboration between objects that lay at the base of the implemented system. Moreover,

a user interface is free from implementation details. Once the interface is understood, the abstractions implied and the output of a system is understood as well. Next, we will take a look at how a user interface is constructed in Quintiq, followed the user interface of our system.

Designing the User Interface (UI) in Quintiq Environment

In Quintiq the implementation is separated from the user interface design. The user-interface is defined in Quintiq Designer (often referred as Designer) which is a part of the Quintiq Environment. As far as types and data structures are concerned (in the Designer), a type is mainly indicated by the term *Component*. Quintiq enables component-based design of the UI. Components are types only defined in the Designer. These types are used to model the visualisation of the *object types* defined in the Bussines Editor (also referred as Editor) part of Quintiq Environment. Unlike the object types, these types cannot be extended with additional attributes, relations, functions or constraints. It is possible to define methods in the user-interface, but the use of methods is restricted only to specific components (i.e. forms, dialogs, reports).

Another difference between object types belonging to the Editor and user-interface components type belonging to the Designer is the support of properties and responses. Properties are used in the Designer to model the appearance and parts of the behaviour of a component. Properties are to user-interface types what attributes are to object types, the major difference being that not all properties are available at runtime. Only published properties are available runtime, these properties can be used in the same way that attributes are used. The published properties can be used to allow the user to configure the interface at runtime. Responses are used to model the behaviour of the user-interface.

Since a component cannot be extended with relations or attributes, it is necessary to be able to refer to entities in the object model (established in the Editor) as well as to be able to (temporarily) store a specific object, for example in order to edit the values for that object. This is achieved in the Designer through the use of *object-binding* and *data-binding*. But we do not get into details.

Creating the user interface is done in a visual way in Quintiq Designer. Three basic steps are required for this:

- Adding components to the deign surface. The components vary from: buttons, forms, labels, lists, to charts, map controllers, or date time selectors.
- Setting initial properties for the components.
- Writing handlers (responses) for specified events.

Routing application's UI

In general, a UI provides means of input through which the user manipulates the system and means of output, through which the system indicates the effects of users' manipulation. The challenge is to keep the interface clear and at the same time simple. For that reason, an operator needs to provide a minimum input to achieve the desired route. As for the output,the proposed UI has a mixed output: comprising of text and visual elements (map of the network used, graph of the variation of speed on a certain road).

The overall structure of the user interface is relatively simple, as shown in the diagram from Figure 4.12. There are four primary elements in the routing application's interface: a *menu*, *buttons*, *forms* and a *timer*. Within the application, a simulation can be run and the routing advice that is given to drivers can be analysed. This simulation results in lot of data and because of that it is important to give the operator a good overview of them. A number of forms with incorporated lists are available. The routing application's

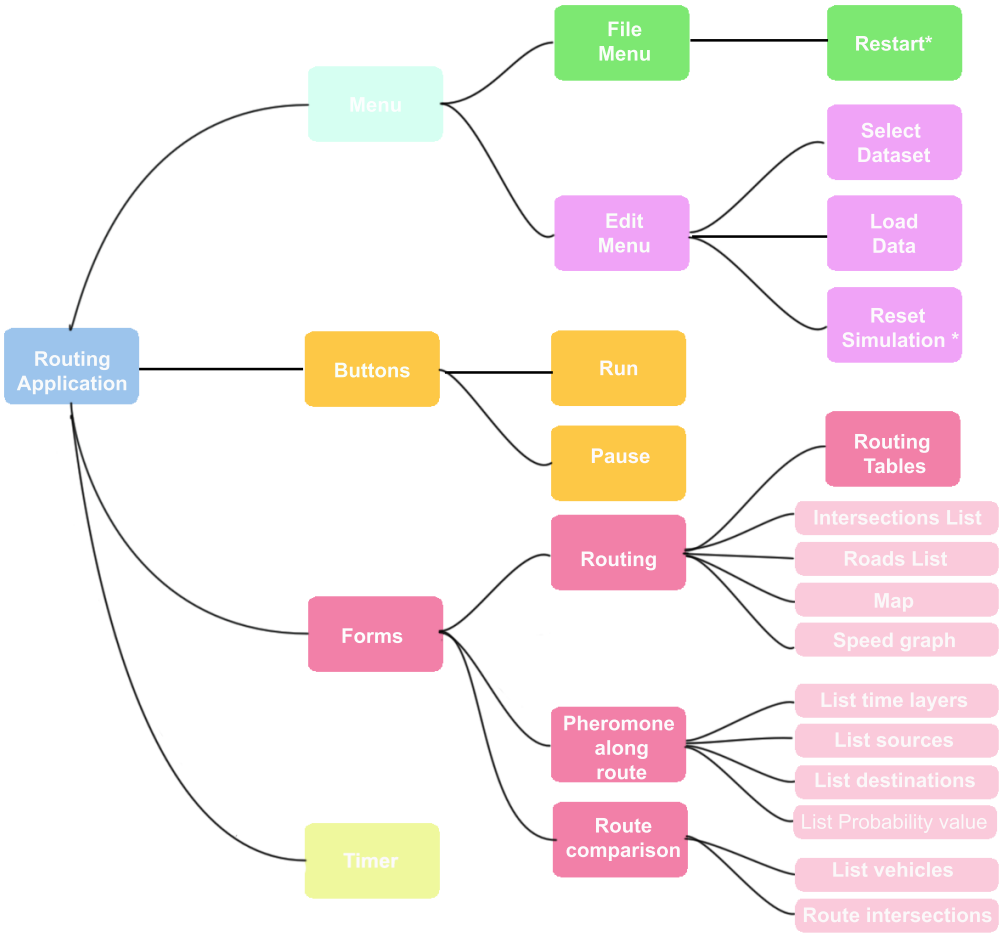


Figure 4.12: The organisation of the user interface components.

The root is the routing application, and the other components are part of it. Certain components can be activated or not depending on the user's preference. These special components are marked with an asterisk symbol.

GUI covers the interaction with the operator and allows the start/the pausing or the reset of the routing simulation, but also the inspection of the routing tables, pheromone accumulated per routes and a route comparison.

For navigation devices or platforms, map based visualisations are quite habitual and successful. We propose a client user interface (UI) that has integrated a map display (Figure 4.13) where the selected network can be observed. The selected highway network is depicted on top of the geographical network with straight lines, even though for each highway the actual distance in considered. Since we didn't yet developed a mobile part for our system, the rendering challenges are not that complex. Panning and zooming are possible. These functionalities were not implemented by ourselves and come with the integrated PTV maps of Quintiq Environment.

But before the dissection of route advice given by this application several steps are imperative. Selecting the dataset, on which we prove that the algorithm developed is functioning with good results, and loading the data are the important things that a user should not forget to do before starting simulation. This is done from the menu as it can be seen in Figure 4.12. All menu actions can be reached using shortcuts.

Except for the *Run* and *Pause* buttons and the *Reset Simulation* menu, the simulation

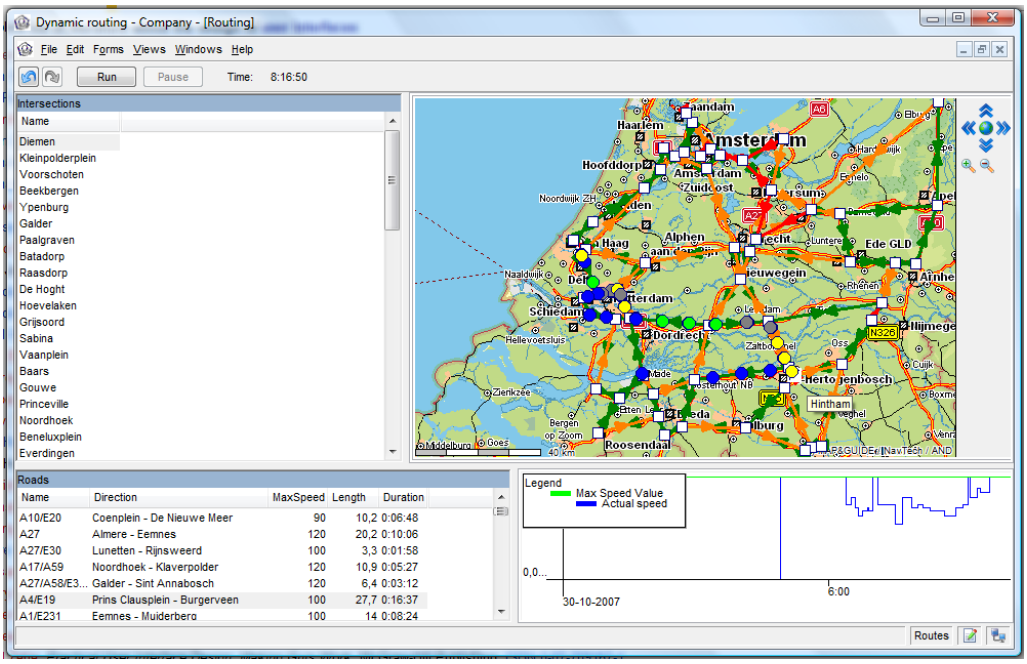


Figure 4.13: The *Routing* Form belonging to the current implementation.

is independent of the user. Every second the *Timer* element, that increments the time count with 1 minute, is called. Actually, the timer simulates the flow of time. The simulation is dependent of this current time. Current time of the application is displayed in the top bar.

The **Routing Form** (Figure 4.13) presents the traffic condition of the entire road network, namely speed variation for every road. Visual and textual cues are used. Every road is rendered by two directional links, but also a list of roads is provided. Different speed levels are also conveyed graphically by changing the colour of the links in the spatial network. One can detect which are the most congested roads in the network by their colour. Red links are most congested, green links are less congested and orange ones are in between as far as the level of congestion is concerned.

After the simulation startup, Pheromone for every possible route can be inspected, (Figure 4.14). Routes are built in the application for every possible pair of nodes in the spatial network. The Form **Pheromone along route** is comprised of four lists (for an example follow Figure 4.14):

1. one where the *source* can be selected (in this example Almere intersection),
2. second one corresponding to the *departure time layer* in which the driver starts his journey (here 8:45),
3. the third one where the *destination* (Coenplein interscetion in this case) is chosen and the *average travel time* (0:32:06) for that destination, taking into account the departure time layer, is given,
4. and the forth in which for the end node of the outgoing link beginning at the start intersection (1), the *probability* is mentioned (0,75 for Muiderberg and 0,25 for Eemnes in this example).

Once the simulation has started, the timer starts simulating the time flow. Due to the modification of the time, vehicles are put into motion according to their route and

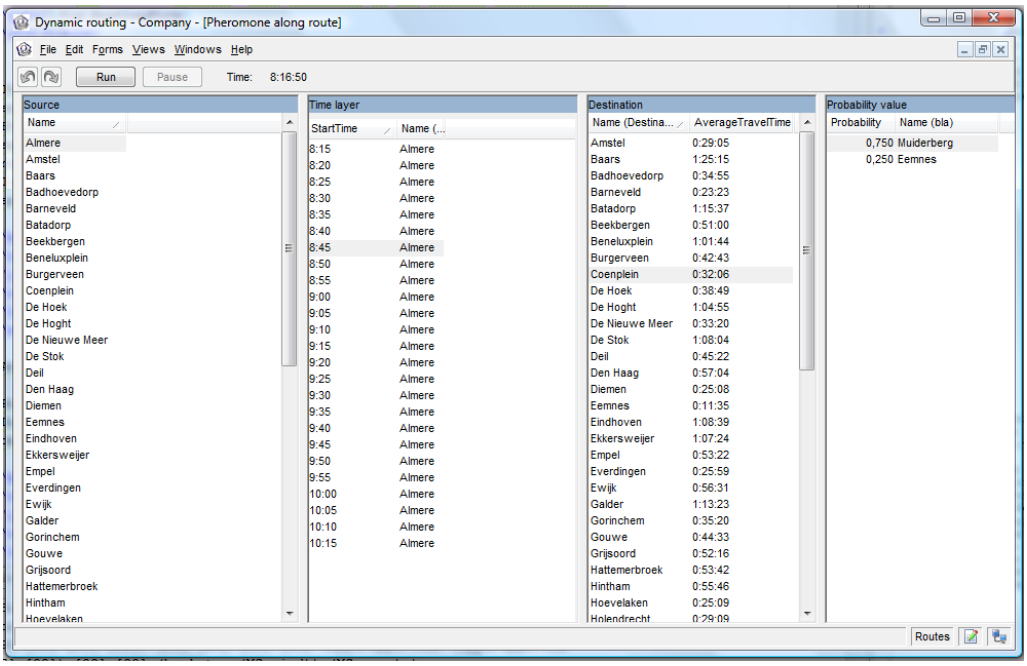


Figure 4.14: The *Pheromone along Route* Form of the current implementation.

dependent or not of the congestion in the network. Ant Based Control is going to use the congestion information for the benefit of the users, while static routing (Dijkstra's algorithm) will not take into account the variation of speed on highways. In the **Route Comparison Form** (Figure 4.15), the operator can inspect the details of a route given by different algorithms. In our case the algorithms are ABC, Dijkstra's algorithm (for convenience called TMC1), Dynamic Dijkstra with updates every 10 minutes (here TMC2) and Dynamic Dijkstra with updates every 30 minutes (TMC3).

The users may also trigger these main forms (Routing, Pheromone along the route and Route comparison) simultaneously and keep them in view all the time.

Appendix contains detailed description of the user interface. Users are guided how to manipulate the routing advice tool through an illustrative scenario with manipulation steps.

By means of the current implementation we have demonstrated the translation of the model that we explained in Chapter 3 into a prototype which provides a test environment for this new version of ABC.

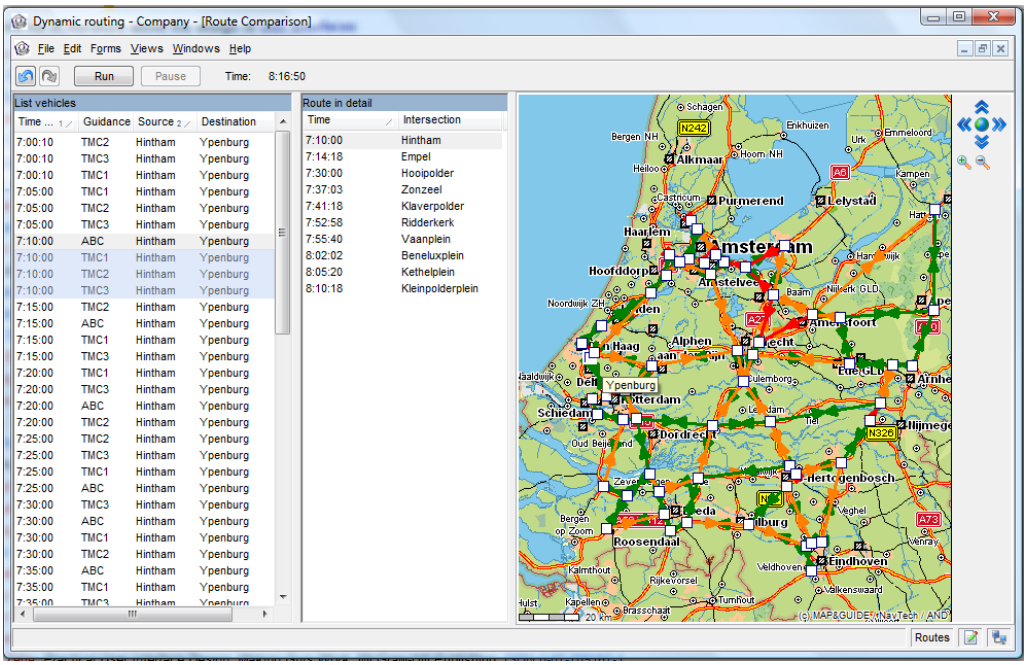


Figure 4.15: The *Route Comparison* Form of the current implementation.

For each vehicle in the list of vehicles going from Hintham to Ypenburg, each with a definite departure time, a detailed route with intersections that one must follow is shown in the *Route in detail* list. By selecting the other 3 vehicles departing at the same hour (highlighted in blue) a comparison of the intersections to follow and the arrival time can be realised.

Chapter 5

Evaluation of the routing algorithm

This chapter describes the manner in which the experiments for testing the routing algorithm were developed together with the detailed results and discussion. The experiments are supposed to highlight the benefits of the proposed approach. It is important to note that the choices made for model design and testing are straightforward.

In the first place, we present the components of the realistic simulation environment: the highway network that constitutes the spatial network model at the base of the simulation and the traffic information collection and filtering. A simulation environment was created to conduct experiments meant to approve or reject the proposed solution design. The main pseudo code parts and the GUI description of the application can be found in Chapter 4. Moreover, simulation it is a very suitable method for experimenting and finding out the shortcomings and strengths of the proposed extension of the ABC routing algorithm. Next, there will be two parallel comparisons with *Static Dijkstra* and *Dynamic Dijkstra*, and an analysis of the algorithm run.

The relevant observations that came out of the experiments are also presented in this chapter.

5.1 Experimental methodology

To begin with, we describe the methodology introduced for the experiments. Experimental settings together with the process of data collection and filtering are detailed in the following sections.

First, there is a need to clarify the difference between *empirical* and *rational methods of evaluation*. Empiricism considers that the truth is inferred from data collected in the world, while exactly the opposite is rationalism philosophy, where deduction of conclusions comes from axioms using proofs. Either one of these philosophies is accepted by the scientific world, but the last one is believed to be stronger. Usually, algorithms that are probabilistic in nature, as is the case in the current thesis, are studied *empirically*. At the same time, theoretical work is also possible when dealing with probabilistic algorithms. There is a slightly disadvantage though that can lead to a difference between the results of the implementation and the predicted theoretical results and it is caused by the fact that in practice simplifications of the implementations are adopted. Nevertheless, in some cases results may depend on the datasets used or on the parameter's selection. In general, an empirical investigation consists of a set of experiments, a set of tests run under controlled conditions to examine the performance of something new.

This thesis is build as an *empirical investigation*. For this reason a methodology has to be specified.

All the results introduced here are coming from simulations. The algorithm has been tested in a network that resembles reality in a great manner. But however, no mathe-

matical proofs concerning the specific properties of the algorithm are provided. In fact, a mathematical treatment require in depth knowledge of the dynamics of vehicular traffic and of the traffic incidents formation processes.

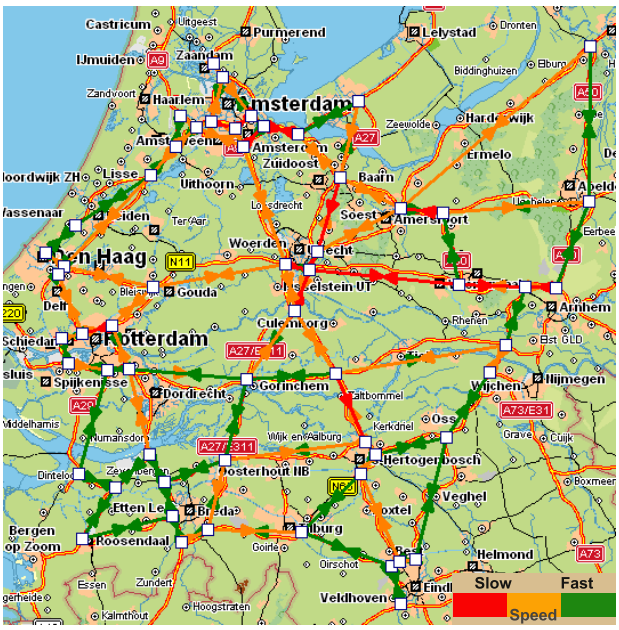


Figure 5.1: Map of selected traffic network used in simulation.

This map illustrates the nodes and links considered in the evaluation of the vehicular routing algorithm proposed.

5.1.1 General experimental settings

Testing and evaluating the routing algorithm requires a detailed representation of a road network geometry. This is the spatial network that we talked about in Chapter 3. As a proof of concept, the experiments will run in the traffic network represented in Figure 5.1. As a network test-bed we selected part of the Dutch highway system belonging to the Ranstad area. The main reason for resuming ourselves to highways (for the time being) is that reported travel time delays are available for these types of roads and not for city centre infrastructure or smaller national roads. For an accurate representation of road distances and coordinates of intersection points we used the Quintiq software which has an integrated GIS system, as mentioned in Section 4.1. The network we built matches the real world system of highways of an important part of the Netherlands and consists of 58 nodes (highway intersections) and 168 directional roads with corresponding link characteristics. Each road is described by the number of lanes and the maximum allowed speed. Some of the real roads are composed of multiple segments with different speed limits and number of lanes. In this cases we choose an average speed limit and a fixed number of lanes.

5.1.2 Data preprocessing and filtering

Often when dynamic vehicle routing systems have been investigated, randomly generated data rather than real-life data was used. But this is not the case since we want to base the results presented here on realistic grounds.

Figure 5.2: Representation of traffic delay from junction to junction.
Diemen to Muiderberg.

Data collection

Traffic delay information that is being used is based on collection of historical information of traffic condition from 2007. It was gathered for several days from ANWB website, one of the most reliable sources of traffic road information in The Netherlands, at every 2 minutes. Historic delays reported by ANWB were collected by R.Sondak [59] for 27 September 2007 and for 4 October 2007. In total, in the interval between 0:00 and 23:58 there were 718 traffic information files. Each file contained for every *hour of the day* and *road segment*, the *minutes of delay* reported by ANWB.

Filtering

For all these 718 traffic information files a filtering was applied for the duplicate data. The empty files and other unnecessary information, like intersections that were outside the selected network range were deleted. Interpolation was applied for missing values. However it is generally known and proven trough this empirical data collection that during a day there are two main moments when the congestion occurs: the morning and the evening.

Creating a historical database of delays

There are a couple of ways to calculate the delay that can appear in addition to the optimal travel time between two intersections, when there is length and/or duration provided. The following computation is required so that the traffic information will match exactly the selected road network. In this sense, there are 4 situations that occur. ANWB text, together with an illustration and the actual computation are going to be presented for each one of them.

Situation 1

A1 Amsterdam to Amersfoort between intersection Diemen and intersection Muiderberg 9 km, delay 20 minutes, length decreases 9 km, Figure 5.2.
This is the simplest case to model. The historical database of traffic information is checked for nodes that match to the current congestion information and the delay is registered in the database. This extra time will be finally added by the algorithm to the normal time needed to travel from A to B.

Situation 2a *A27 Almere to Gorinchem between Hilversum and Utrecht-Noord 3 km, delay 1 minute, length decreases 3 km, Figure* 5.3.
The calculation is done in the same way as in situation 1, since the delay is between 2 adjacent nodes in the selected network.

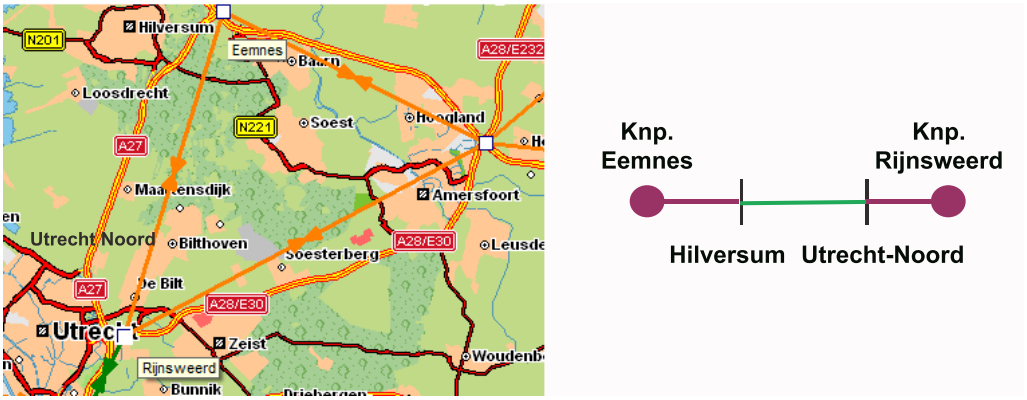


Figure 5.3: Traffic delay on one segment between adjacent junctions (Point A and B).

Situation 2b *A1 Amsterdam to Amersfoort between Naarden-West and Bussum 7 km, delay 7 minutes, length decreases 7 km, Figure* 5.4.
A1 Amsterdam richting Amersfoort tussen Laren en Eemnes 4 km, Vertraging tot 2 minuten 4 km. 5.4.
The delay time from exit 1 to exit 2 is summed up with the delay time from exit 3 to exit 4, since the congestion is between two adjacent junctions as coded in the delay database. The delay between Knp. Muiderberg and Knp. Eemnes will be then 9 minutes.

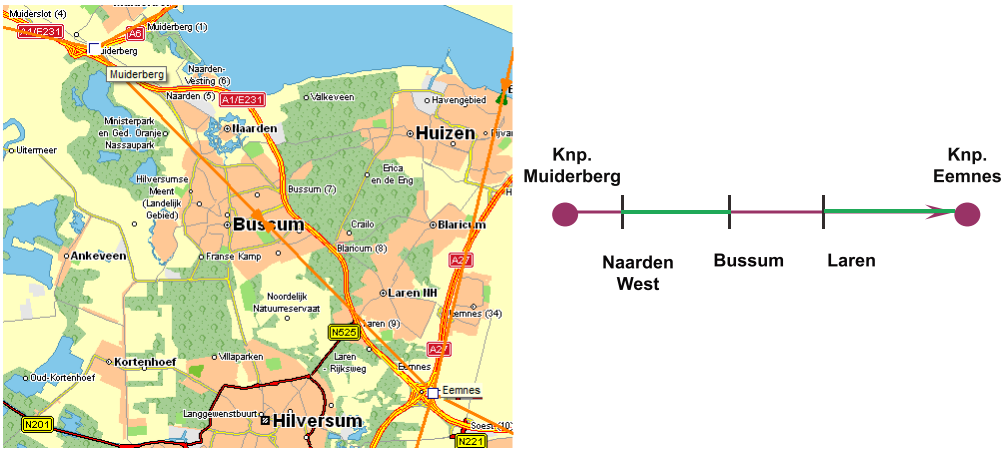


Figure 5.4: Traffic delay on two segments between adjacent junctions (Point A and B).

Situation 3 *A27 Utrecht to Gorinchem between Houten and Lexmond 9 km, delay 15 minute, length decreases 9 km, Figure* 5.5.
In this case the delay between junction A and junction B is calculated proportionally with the distance between A to B reported to the distance between A to C. This is also applicable to the delay between junction B and C. The following relations illustrate better the situation.

$$delay_{AB} = \frac{l_{AB}}{l_{AC}} \times delay_{AC}, \text{ where } l_{AB} \text{ and } l_{AC} \text{ is the length of AC and AB,} \quad (5.1)$$

$$delay_{BC} = \frac{l_{BC}}{l_{AC}} \times delay_{BC}, \text{ where } l_{BC} \text{ and } l_{AC} \text{ is the length of BC and AC.} \quad (5.2)$$

After some calculations the delay that resulted was:

$$delay_{AB} = \frac{9.8}{19.7} = 7min\ 27s,\ delay_{BC} = \frac{9.9}{19.7} = 7min\ 32s. \tag{5.3}$$

Situation 4 *A1 Amsterdam to Amersfoort between Soest and Bunschoten 3 km, delay 7 minute, length decreases 3 km. A1 Amsterdam to Amersfoort between Eembrugge and intersection Hoevelaken 3 km, delay 7 minute, length decreases 3 km , Figure* 5.6.
The calculation is a combination between situation 2b and situation 3, when we added up the traffic delay information. However, we remove the overlay point in order to avoid duplicate data.

$$delay_{AB} = delay_{CE} + delay_{EB}. \tag{5.4}$$

Finally, the delay between knp. Eemnes and knp. Hoevelaken will be: 10 minutes and 1 second.

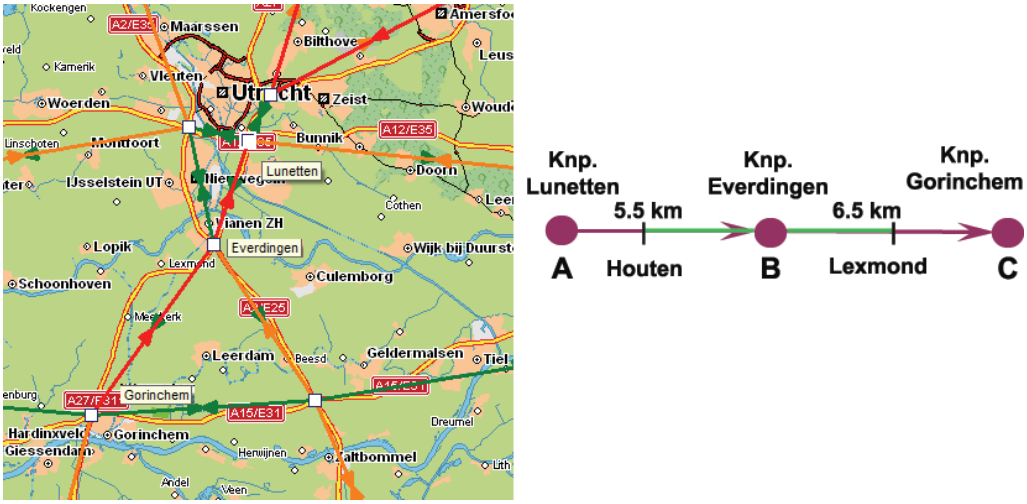


Figure 5.5: Traffic delay from junction to junction (A to C) with intersection in between.

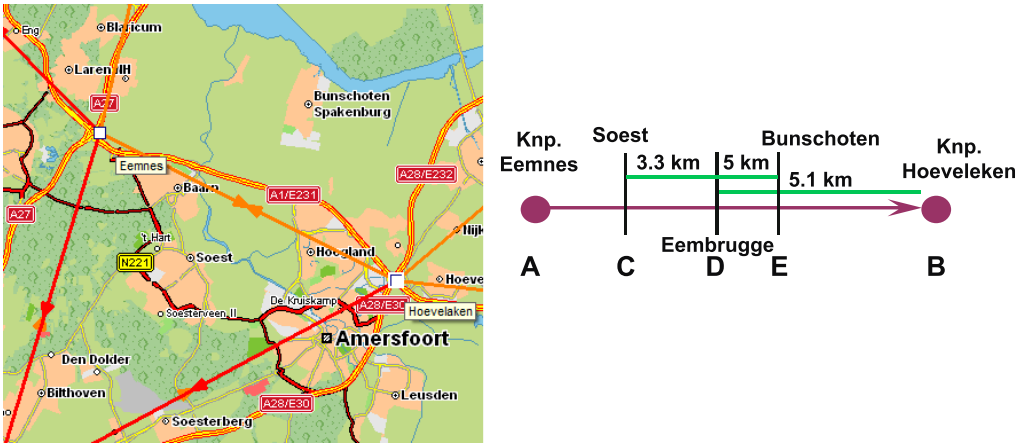


Figure 5.6: Traffic delay from two traffic information where the exits/junctions overlap.

5.1.3 Routing algorithms used for comparison

The performance of the new AntNet is compared to that of the following routing algorithms which are taken as representative for the problem under consideration in the field of static and dynamic driver routing.

Static Dijkstra's algorithm has the same structure as in Section 2.2.

Dynamic Dijkstra (DD) with updates is a modified version of the Static Dijkstra's algorithm. DD considers that updates of the *actual free flow speed* v_{ij} are received either at every $\Delta t = 10\ min$ or at every $\Delta t = 30\ min$. The cost of every road segment is the travel time, computed as in Algorithm 7, and not the road distance as in the original Static Dijkstra. We have to chose the traveling time as the cost of an edge because the routing process should offer the fastest path between two intersections.

The updates that are provided to DD are also simulated based on historical data measurement from ANWB.

Algorithm 7 Pseudo-code of procedure *GetTravelTime* for Dynamic Dijkstra (DD).

This procedure implements the travel time estimation methodology for two successive nodes i and j for Dynamic Dijkstra.

$GetTravelTime(i, j, t_i, v_{ij}(t))$
  $\{i$ - current node, j - successor node of $i\}$
  $\{t_i$ - current time at node $i\}$
  $\{t$ - the last update time$\}$
  $\{v_{ij}(t)$- the free flow speed from i to j at time $t\}$
  $L \leftarrow d(i,j)$ $\{d(i,j)$ distance between i and $j\}$
  $k \leftarrow 0$ {start the iteration with the first time interval of that road}
  while $t_i \in I_k$ **do**
    $t_i \leftarrow t_i + \frac{L}{v_{ij}(I_k)}$
    $k \leftarrow k + 1$
  end while
  return t_i

5.2 Path selection: adaptivity test

In this section the adaptivity of the new ABC algorithm to changing traffic conditions is going to be tested and analysed. The mechanism that underlies the adaptivity of the solution construction is the ant's permanent exploration through the environment. As was stated in Section 2.3 and in Section 3.3, ant agents have the capability of sensing traffic speed variations that occur due to high flows of vehicles. This ability of the ant agents influences the route selection procedure.

Further on, to illustrate the adaptivity of the path selection, a particular case will be considered in the conditions of the network described in Section 5.1.

's Hertogenbosch-Den Haag (intersections Hintham-Ypenburg)

If a driver decides to start his journey in Hintham (an intersection close to 's Hertogenbosch) and wants to arrive at Den Haag (intersection Ypenburg), several of path possibilities that connect these two intersections have to be weighted. Of course, any static algorithm (in our case the Static Dijkstra) chooses to guide the driver via the route that has the shortest distance. This route will always be selected because Dijkstra's algorithm is not aware of the fluctuation of speed during a day. The shortest path

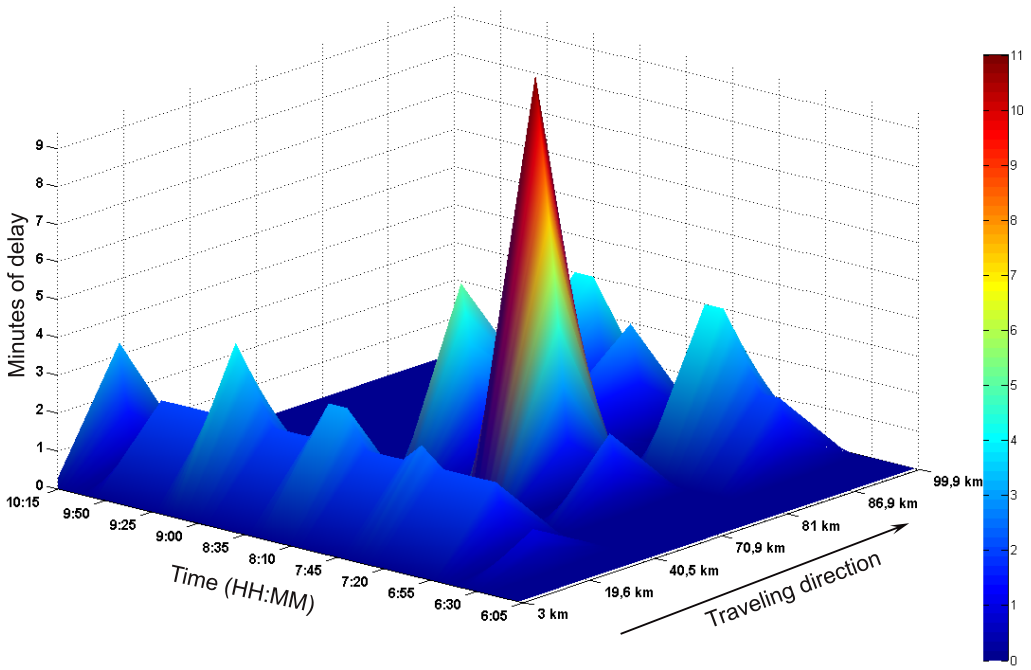


Figure 5.7: The delay encountered by drivers on Route 2 (see TableB.3).

The delay is based on traffic information from 27.09.2007 when starting the journey in Hintham and arriving at destination Ypenburg.

in the case of the current source and destination is presented in Table B.3. Drivers who follow this route will encounter the delays in Figure 5.7. One has to be aware that the shortest route is not always fastest route in time. If Dynamic Dijkstra with updates at every 30 minutes (DD 30 min) is used in selecting the route, then the route chosen is different (Table B.4) and is 0.7 km longer. Dynamic Dijkstra with updates every 10 minutes (DD 10 min) reacts faster that Static Dijkstra and than DD 30 min to modifications of the free speed. Differently said, DD 10 min reacts to the delays that appear in addition to normal travel times. This DD alternates between two routes that are detailed by Table B.5.

Traffic fluctuations undoubtedly make a difference in route selection for an adaptive routing algorithm. This is best illustrated by the new ABC, which fluctuates the path selection between 6 different routes, precisely those in Table B.2. In percentages, the distribution of routes selected is the one in Table B.1. The same distribution over the time period 7:00-12:00, 27.09.2007 can be found in Table B.6.

5.3 Traveling time: effectiveness analysis

The experimental results that follow will show that the modified ABC clearly outperforms all the considered competitors.

Eindhoven-Amsterdam (intersections: De Hoght-Muiderberg)

First the A2 motorway, which connects Eindhoven to Amsterdam. It is the most congested corridor in The Netherlands. Here we selected for a detailed discussion the route between the junctions De Hoght, situated at west of Eindhoven, and Muiderberg, positioned east of Amsterdam (Figure 5.8). The shortest path has a length of 104 km. There are two main alternatives available. One is to go via Tilburg and follow A27 to

Utrecht (130 km). The second, also 130 km long and presented in the figure, is via Nijmegen and Arnhem using A50. For the segment between Utrecht and Amsterdam, the main options are via A2 direction ring Amsterdam, or via A27 and A1.

Figure 5.8: The shortest versus the fastest route from De Hoght to Muiderberg at 9:30.

The route in light colour is the shortest and the most direct one, while the route in a darker colour is the fastest at 9:30.

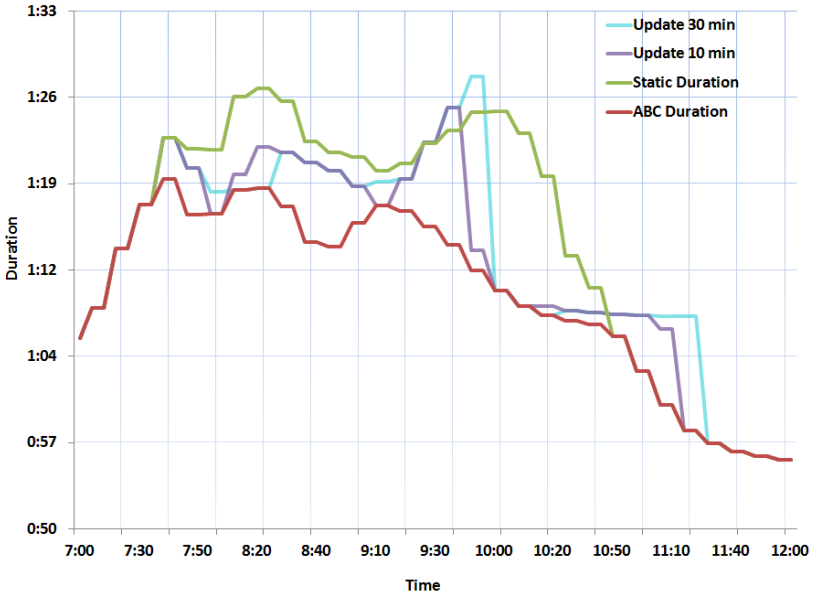


Figure 5.9: Traveling time from De Hoght to Muiderberg for 7:00-12:00, 04.10.2007.

In Figure 5.9 is shown how the travelling time between these two locations has been oscillating during the morning. Between 7:40 and 10:40 the ABC algorithm suggested different faster routes than the static one (Dijkstra). Around 10:00 the ABC suggestion via Nijmegen was 15 min faster despite the additional 26 kilometres. The results of the

guidance using the current traffic updates were somewhere in between. Increasing the frequency of receiving the traffic information from every 30 minutes to every 10 minutes, didn't show much difference. It can also be observed that for a while, around 9:50 and 11:00, these suggestions were actually slower than the static route. This is because, the alternatives were calculated only using the actual state of the congestion, and the future evolution of the traffic jam was not considered. This was not the case for the routes calculated by the ABC algorithm.

's Hertogenbosch-Den Haag (intersections: Empel - Prins Clausplein)

For the second particular case we chose as starting point the motorway intersection Empel situated on the ring of 's Hertogenbosch not far away from the Quintiq office (Figure 5.10). The destination is Prince Clausplein intersection. It is situated north from Delft close to The Hague. The shortest way (82 km) is via Utrecht following A2 and A12. But again, many alternatives are available, like going via Waalwijk on A59 then A16 to Rotterdam or via Gorinchem on 15. Also Rotterdam can be passed via east or western ring. The suggestion in the figure is 98 km long.

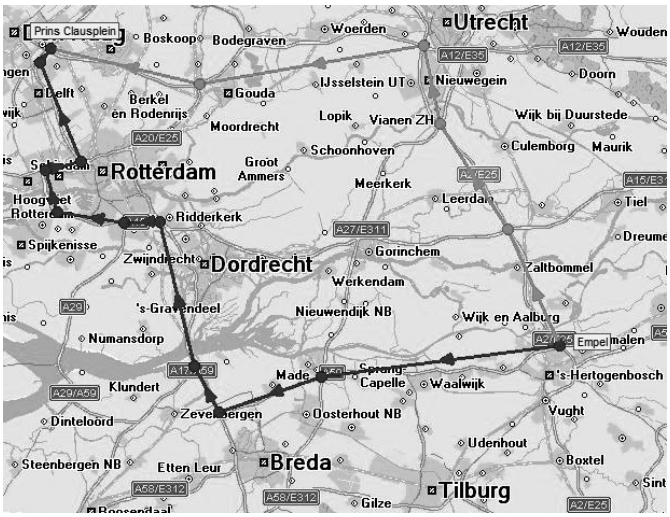


Figure 5.10: The shortest versus the fastest route from Empel to Prins Clausplein at 8:30.

The route in light colour is the shortest and the most direct one, while the route in a darker colour is the fastest at 9:30.

On this route the ABC suggested better paths from 7:00 to 9:40 (Figure 5.11). At 7:50 the highest difference from the static route was recorded: 13 minutes and 39 second. Again we can observe that using the current navigation is not always efficient and sometimes the suggested path proves to be longer in time than the direct route.

5.4 Pheromone evolution: stability test

The stability of the system, or the degree to which the system remains in the same state, is going to be tested in this section. We expect that the ant-like agents will continue to move and update the probability tables.

Probability tables are very important since they constitute the basis for route selection. At every intersection, the highest probability determine the next intersection to be selected from the set of neighbours that intersection has. Kroon [46], stated that a routing system is stable as long as the highest probabilities remain the highest for the entire time horizon of the simulation. Moreover, Kroon was also the one who proposed

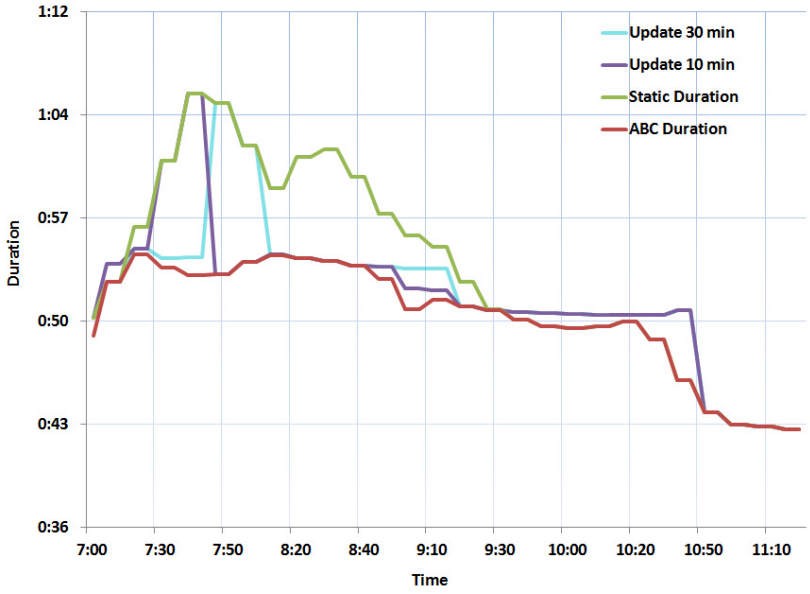


Figure 5.11: Traveling time from Empel to Prins Clausplein for 7:00-12:00, 04.10.2007.

to measure the system's stability by computing the instability. Instability counts the number of times the highest probability switches to an alternative node. It is argued that neither stability and instability alone can account for the global performance of a routing system.

Figure 5.12: The neighbouring intersections of Ridderkerk.

Figure 5.13: The neighbouring intersections of Empel.

A new experiment based on the network in Figure 5.1, for the route *Empel-Ypenburg* was performed. The simulation covered data from 04.10.2007, but just the results of the morning peak (7:01- 9:00) are reported. Only one route is considered: starting in

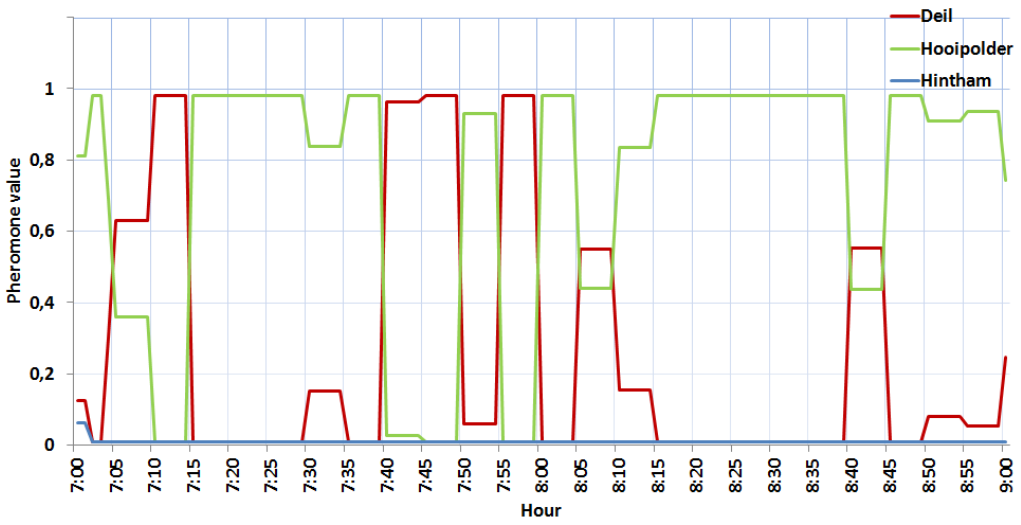


Figure 5.14: Probabilities in node Empel for the route Empel to Ypenburg.

Empel and ending in intersection Ypenburg. The route connects two important cities in The Netherlands: 's Hertogenbosch and Den Daag. The pheromone values for the intersections: **Empel** and **Ridderkerk** are plotted in Figure 5.14 and respectively Figure 5.15. Intersectin Empel is at the start of the road and has the neighbours depicted in Figure 5.13. Being close to the start intersection, means there can be available several alternative routes. It is like being at the source of a tree data structure. On the other hand, intersection Ridderkerk is very close to the destination intersection Ypenburg. For its neighbours in the spatial network, see Figure 5.12. In intersection Ridderkerk the alternative routes have already been exhausted, and it is like being close to the leaf of a tree data structure.

When the current intersection is Empel, the instability of the system has the value 10. The value is quite high. And one explanation for this may be: the route will change the direction more often at the source than when it is closer to the destination. The routing algorithm switches the highest probability value between 2 of its 3 neighbours, specifically: Hooipolder and Deil. If the third neighbour Hintham, is selected would mean that the driver will get further away from the destination and not closer. The graphic in Figure 5.14 reflects that intersection Hooipolder is most often preferred by the ants, while the probability of choosing Hintham as the next intersection is almost zero. This probability does not actually reach zero as a consequence of having a reinforcement value which supports exploration.

But when the current intersection is Ridderkerk the value of the instability is only 3 (see Figure 5.15). Neighbours Terbregseplein and Vaanplien are the favourites of the ants.

The ant-like agents determine major changes in the pheromones only 3 times. Whether and how are the changes of the pheromones related to the traffic condition change, has yet to be studied. There are to few data samples to make a generalization. Maybe the highest pheromone level should not drop so abruptly, but then the reinforcement parameter needs some adjustment. Additionally, after 8:00 the pheromones have a high degree of stability.

Closed road case

Another test was performed for source: Empel and destination Ypenburg. The outgoing link Empel-Hooipolder was closed between 7:30-8:15. Due to this sudden modification of the road graph, the probabilities have changed their shape to the ones in Figure 5.16.

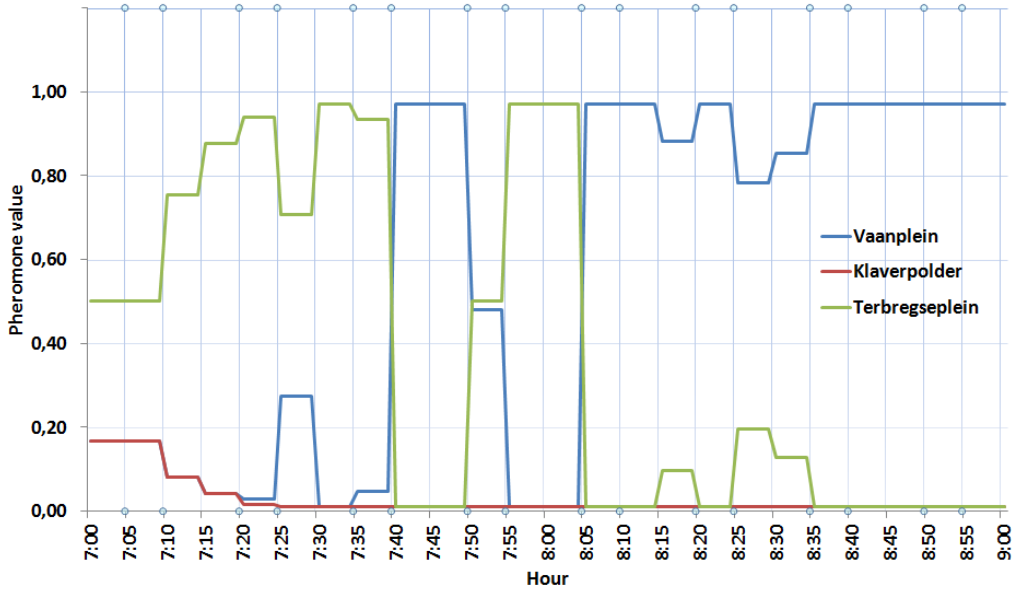


Figure 5.15: Probabilities in node Ridderkerk for the route Empel to Ypenburg.

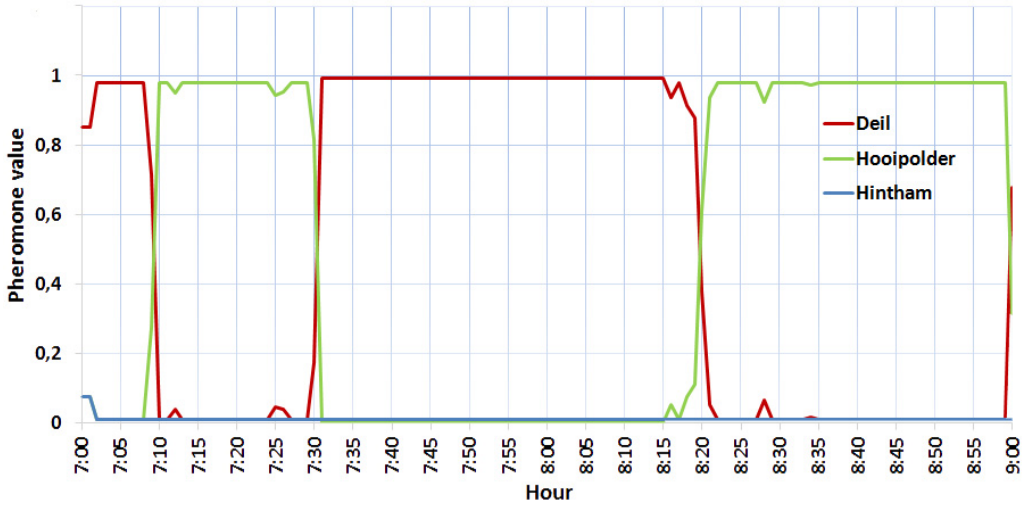


Figure 5.16: Probabilities in Empel in case Empel-Hooipolder road is closed.

The road was intentionally blocked from 7:30 to 8:15. Probabilities consider that destination is Ypenburg. This is based on historical data from 04.10.2007.

If Figure 5.14 is compared to Figure 5.16, one can observe the fluctuation between the alternative neighbours Deil and Hooipolder during 7:30-08:15 have disappeared. We expected that this will happen, because now the only viable neighbouring intersection to choose is Deil.

5.5 Global improvement

Figure 5.17 displays for the three 'smart' strategies selected (ABC, DD 10 min update, DD 30 min update), the percentage of improvement compared to the static routes. In average for 50% of the routes, faster alternatives were found by the routing system. A route is considered improved when it is faster than the initial route proposed. Nevertheless, one should be aware that for the results in Figure 5.17, only the best difference obtained during the simulation time was recorded. The number of the deteriorated routes was very low and depends on the granularity of the time intervals used by these three algorithms. The results obtained using ABC algorithm indicate that 53% of the routes were

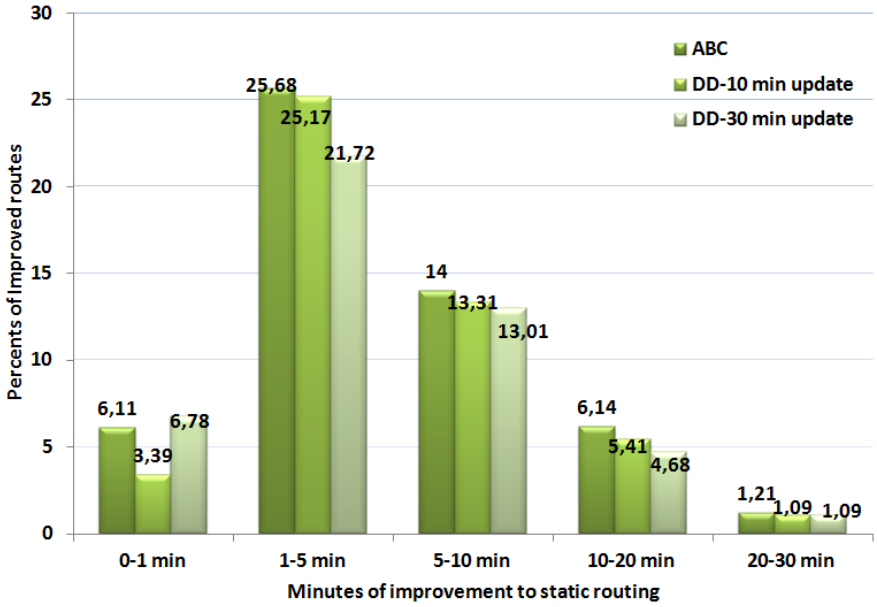


Figure 5.17: The difference between ABC and DD in percentages.
Results are obtained using traffic data from 04.10.2007.

improved. From this 53%, for 6.11% of the cases an insignificant improvement of less than one minute is gained compared to the static path. Only 26.68% of routes suggested by the ABC were up to 5 minutes faster than the static ones. An improvement of 5 to 10 minutes was noticed in 14% of cases and of 10 to 20 minutes in 6.14% of the situations. For 1.12% of the routes the improvement was up to 30 minutes. If we consider that The Netherlands is a small country and most of the selected routes are shorter than 150 km, then using the dynamic routing system is a real benefit.

5.6 Analysis of the algorithm run

5.6.1 Average age of living forward ant

Experiment settings. A simulation was implemented from 7:00 AM on 27 September 2007 (simulation time) to 11:00 AM of the same day (simulation time) in the network depicted in Figure 5.1. In this simulation only vehicles traveling from Hintham to Ypenburg were considered. Moreover, in this particular case ants were launched from every node to any other node, for each car using ABC and from the selected source.

Results. Figure 5.18 represents a graphic showing the average age of the living forward ant at every 5 minutes. The average age is obtained for all the ants launched.

An important observation is that at some moment in time, there are good ants and bad ants. Bad ants will have cycles in their paths and will be killed later on.

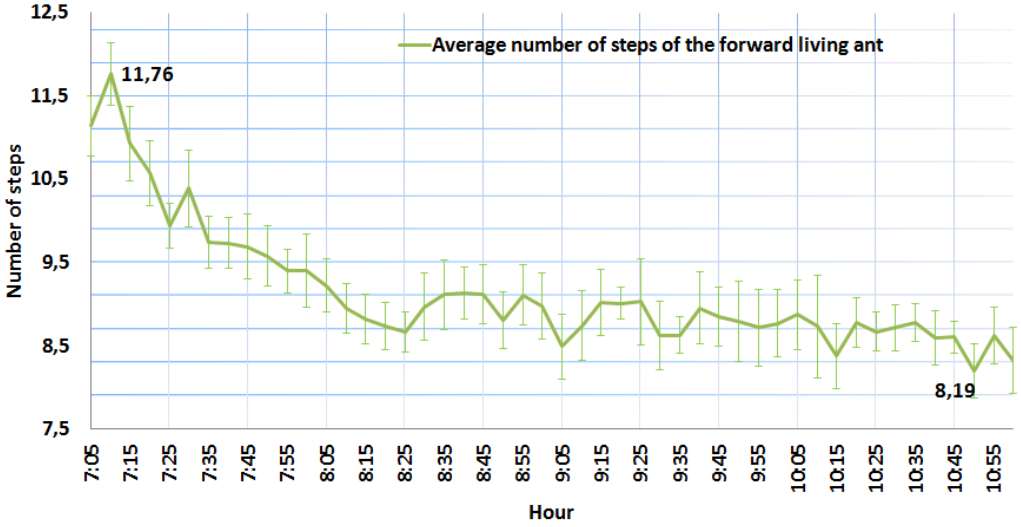


Figure 5.18: Average age of living forward ant for route Hintham - Ypenburg.

The results come form a simulation over the interval 7:00-11:00 AM 27.09.2007, at every 5 minutes. The errorbars account for ten rounds. The simulation takes place in the network form Figure 5.1. The travelling time suffers from delays that have been reported by ANWB Verkeer organisation for this particular day.

From ABC and AntNet we know that backward ants have the same age as forward ants and they deposit pheromone along the route. The normal living forward ant makes on average approximately 9 steps in the experiment. In the beginning of the simulation the number of ants is around 11, above average, but this because the ants follow unknown routes. The backward ant will update all probabilities for the nodes in the stack of the forward ant. But backward ants update also the subpaths. So the backward ant will make on average 45 updates at every 10 seconds in this particular case. The frequency of the updates can, however, be changed.

But nevertheless, the number of steps a forward ant performs depends on the number of hops in the possible routes that are considered by ABC. Table B.2 illustrates also the number of segments for all the six alternatives that were considered by ABC when starting in Hintham and heading for Ypenburg. From this table, the average number of steps results to be also around 9, which matches the previous number obtained empirically.

5.6.2 Number of forward ants

Experimental settings. A simulation over the interval 7:00-11:00 AM on data from 27.09.2007 was created for the network in Figure 5.1. The average number of forward living ants at every 5 minutes was recorded. For the average 10 runs were performed.

Results. The distribution of the number of forward ants can be analysed in Figure 5.19. Given that we launch ants for every pair of nodes in the network at every 5 minutes and also launch ants for each vehicle that is simulated, the number of ants varies from a minimum of 114 to a maximum of 140 in the case of the specified network.

As on can see, the simulation is not computationally intensive from the point of view of how many ant-like agents are needed.

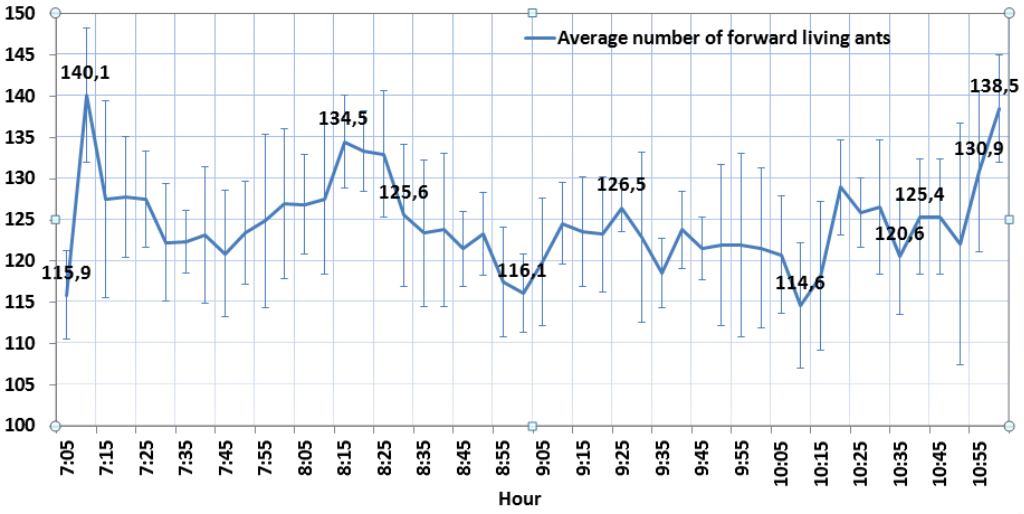


Figure 5.19: Number of ants for the period 07:00 AM and 11:00 AM of the simulation.

The simulated day is 27.09.2007 and the number of ants is given from 5 to 5 minutes. The simulation takes place in the network from Figure 5.1. The travelling time incorporates delays that have been reported by ANWB Verkeer organization for the mentioned day.

Chapter 6

Conclusions and Future Work

6.1 Summary

The main goal of this thesis has been the adaptation or the design of an algorithm with focus on adaptive routing with respect to travel time variation for road drivers. During our discussion of the related literature and of the approaches that have been developed by other researchers, we found swarm intelligence a promising field for dynamic routing. The purpose was to provide to drivers routes that avoided traffic congestion: in the present and/or in the future.

In our case, the literature survey meant learning from the state of the art of routing, examining it critically an trying to perpetuate the best aspects in a proper way. Through the first part of Chapter 2 we discussed the biological context of inspiration and presented the properties of some practical algorithms that were applied to static and dynamic problems in a hierarchical and non-hierarchical way. Even though the review of the related literature was theoretical, we provided also the main characteristics of ant and bee-hive classes of swarm intelligence algorithms. In Chapter 3 we presented there general design aspects for a routing system, the resources utilised in the current model, the characteristics of the algorithmic approach to routing. This chapter also included a travel time estimation methodology for road segments and assumptions that we have supposed as far as vehicular traffic is concerned. We have improved ABC to make it suitable for adapting to traffic conditions in a road network. Further on, Chapter 4 corresponds to the implementation details of this new algorithm. The adaptivity, the effectiveness and the stability of the proposed algorithm has been validated. Global improvement realised by the new algorithm in simulation studies is also discussed in Chapter 4.

6.2 Reached goals

The ABC inspired by ACO metaheuristic is undoubtedly the best choice for dynamic and distributed routing (distributed from the point of view of the agents), partially due to its popularity and partially due to its modular and simple structure based on ant-like agents. Such a heuristic has promised to generate near-optimal solutions based on the interaction of simple agents that move with infinite speed and that independently construct solutions according to a stochastic decision policy. The agents of ABC use a technique of collective learning based on locally dependant pheromone variables. Nevertheless, we believed that this ant-like agents could benefit from the traffic delay information in order to route drivers in a road network, taking into account the actual situations of congestion. By incorporating a dynamic travel time estimation methodology, our supposition has come true.

The reached goals of the thesis are briefly discussed in the lines that follow, on a goal

by goal basis.

Related work review

Indeed the related work, that was covered and summarised in Chapter 2, helped us in obtaining sufficient theoretical knowledge which was used to finalize the objectives set in thesis. The review of dynamic routing issues, does not account for all the possible questions and alternatives that may appear when developing a routing system. The literature in the field of routing is extensive and it would have been impossible to cover everything. However the related work description was concentrated in a great measure on the *state of the art* routing techniques offered by swarm intelligence: ant and bee-based routing.

The dynamic routing problem is the neglected relative of static routing. In a few words, dynamic means awareness of the *spatio-temporal* variation of the road segments delays over a specified period of time. While conducting the literature survey we found little methodological work that focuses on implementing the ability to adapt to traffic congestion into various models and algorithms. After reaching the conclusion that this aspect of routing is so little considered, we have decided to work towards covering this gap.

Algorithm design and adaptation

The second goal, namely the design or adaptation of a routing algorithm that would offer the shortest path was reached in a certain measure by the model in Chapter 3. The ant routing, as particular instance of the stigmergic multi-agent modeling has been transformed into a system prototype that produces the shortest route in time and takes into account the dynamic network conditions. The routing system is providing path advice on an intersection to intersection basis. The routing system was implemented as centralized for the time being because technical decentralized architecture was not available. But plans are to move to a decentralized system, because the algorithm permits it.

In this Chapter 3 we used a *time extended* graph to represent the *spatial* and *temporal* dimension of the traffic network. A time extended graph has a tridimensional structure, where the third dimension is occupied by the travel time delay of the roads in the spatial road network graph. Afterwards, ant-like agents were designed to find the fastest path between two intersections in the time-expanded networks. These agents are collecting delay information and can sense congestion when it happens. One ant can travel several layers into the future so that it estimates the average travelling time for any road. Through the collaboration of many ant-like agents that store their gathered information locally, an optimal route is computed.

However, there are some things that were hard to accomplish. One of the them is route computation based on expected traffic streams (determined also using ant-like agents).

Implementation of the running prototype

Implementation of the suggested model was covered in Chapter 4. This implementation can be supplemented with several databases or several knowledge bases because it has the ability to exploit existing traffic information sources. UML models and pseudocode based description offer more details about the implementation.

What is more, the system was designed less considering the driver but more like a simulation where an operator can analyze aspects of the algorithm.

Analysis and validation

The new ABC (ABC with integrated actual travel time route choice) has showed that is capable of competitive routing under dynamic and static conditions. In Chapter 5

we have chosen two algorithms for comparing the dynamic ABC algorithm against, the static routing of Dijkstra and a Dynamic Dijkstra. The main interest was to perform the testing under realistic assumptions and for a couple of scenarios and routes.

The understandings of the workings of the algorithm, was realised through several experiments. The new ABC selects routes from the whole range of possible alternatives between a source and a destination. Its competitors have a less limited set of route alternatives due to the lack of awareness of traffic conditions. The effectiveness analysis that was performed in Section 5.3 proved that an important reduction of 53% of the traveling time was achieved. In other historical traffic conditions, the improvement could be possible bigger.

From the experiments performed we learnt that ABC is not computationally intensive and that it has pheromone stability when the pheromone tables are properly trained.

6.3 Description of the findings

The time has come to sum up the findings of this project.

In general, the provision of traffic information could offer benefits to travelers looking to have optimal travel times. In the experiments performed in this project it was proven that ABC switches between routes when congestion is sensed by the ant agents (see Appendix B).

Moreover, it was attested that incorporating traffic information in the processing of ABC, does make a positive difference for drivers. An important reduction of travelling time was observed (see Figure 5.17) especially during peak hours. From the total number of routes simulated, a proportion of 53% were improved. In 6.11% of the cases there was an insignificat improvement of the travel time with up to a minute, while for 26 % of the situations the improvement was growing up to 5 minutes. Greater travel time improvement of 10-20 minutes, was visible in 14% of the routes, and only 6.14% were over 20 minutes. These proportions are on the limit of verifying or rejecting the proposed routing algorithm extension (ABC extension) presented here. It has to be taken into account, nevertheless, that in the considered network, the majority of traffic delays is less than 5 minutes per road. In addition, some detour routes have sometimes similar lengths (see Table B.2). The new ABC still provides some routes that are longer or equal to the routes calculated by other algorithms used in comparison. So a great improvement can't be entirely and objectively justified in this historical traffic situation. The results are network specific and depend on the traffic information selected.

But, in general, the experimental results showed that the modified ABC outperforms its competitors with faster routes. Two scenarios were analysed in this respect for two of the most congested corridors in The Netherlands: Eindhoven-Amsterdam and 's Hertogenbosh-Prins Clausplein.

Another experiment, analysed the pheromone evolution at an intersection. This also showed that ABC for time-extended networks is *able to adapt to changing traffic conditions* and *to exploit the traffic delay* in a road network so that the optimal route (fastest) is given to a self-interested driver.

Last, but not least, an experiment gave snapshots of the average age of living ant and of the number of ants running in the simulation. The simulation proved, from this perspective, to be not computationally intensive.

6.4 Future work or possible extensions

There is still a lot to do in the field of routing, but this project has covered only a small part of the problems met by travellers during their everyday trips. If we were to cluster

together in separate groups the improvements from which the routing problem could benefit even more, there would be two main categories:

- **Possible improvements to current approach**

  Even though, the integration of actual cost estimation of a route has proved to be beneficial for travel time reduction we are far away from an ideal situation. A better parameter tunning and analysis (of parameters like η, α) may bring future benefits, if we were to extend the work of this project. Another aspect, that has to be thoroughly considered is how far in the future layers can ant agents be launched. The reader has to be aware that we do not know how the travelling time evolves when using the new ABC for groups of travellers. We have only considered how is the individual traveller affected. Last in this category, but not least, we would be interested in integrating actual traffic demand into the vehicle simulator that we have built.

- **Other directions**

  The problem of traffic routing is vast and complex. Therefore, many aspects of routing were not considered in this thesis due to the lack of time. If we were to extend the current project into other area of research, traffic assignment and control will be approached first. But new ideas and research fields bring other problems. How is the trip distribution of multiple selected routes between the same origin and the same destination going to be split among the vehicle drivers that use the same algorithm for route computation?

  Secondly, we would consider travel time or speed forecasting. Implementing a mechanism of actual traffic speed prediction would have to consider how far in the future can this variable be predicted and what are the best techniques that could be used. A step forward in improving the situation of the current traffic congestions is to build a real-time traffic prediction system.

  Other direction that has to be considered, is the fact that the routing problem is actually a multi-parameter/objective problem. Different, simultaneous and somehow contradictory conditions can be imposed by the driver upon requesting a route. The question would be then: based on which model can a system ground its routing decision?

  Finally, a regular traveller can switch between different modes of transportation. It would be maybe useful to integrate also alternative modalities into a routing system.

But of course, the above mentioned directions of improvement can represent the work of entire groups of scientists over an extended period of time.

Bibliography

[1] S. W. Ambler. *The Object Primer: Agile Model Driven Development with UML 2*. Cambridge University Press, 2004.

[2] D. Angus and C. Woodward. Multiple objective ant colony optimisation. *Swarm Intelligence*, Volume 3: 69–85, March 2009.

[3] E. Anshelevich and S. Ukkusuri. Equilibria in Dynamic Selfish Routing. In *SAGT 2009: Proceedings of the 2nd International Symposium on Algorithmic Game Theory*, volume 5814 of *Lecture Notes In Computer Science*, pages 171–182, Berlin, Heidelberg, 2009. Springer-Verlag.

[4] B. Baran and R. Sosa. A new approach for antnet routing. In *Proceedings of the 9th International Conference of Computer Communications Networks IEEE ICCCN-2000*, Las Vegas, USA, 2000.

[5] T. Bellemans. *Traffic control on motorways*. PhD thesis, Katholieke Universiteit Leuven, Department Electrotechnik, May 2003.

[6] C. Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4): 353 – 373, 2005.

[7] C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 34(2): 1161–1172, 2004.

[8] C. Blum, A. Roli, and M. Dorigo. H-ACO: The hyper-cube framework for ant colony optimization. In *Proceedings of MIC'2001-Metaheuristics International Conference*, volume 2, pages 399–403, Porto, Portugal, 2001.

[9] J.L. Boehle. City-based parking and routing system. Master's thesis, Erasmus University Rotterdam, April 2007.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence, From Natural to Artificial Systems*. Oxford University Press, 1999.

[11] E. Bonabeau, F. Henaux, S. Gurin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in Telecommunications Networks With smart Ant-Like Agents. In *In Proceedings of IATA 1998, Second Int. Workshop on Intelligent Agents for Telecommunication Applications. Lectures Notes in AI*, volume 1437, pages 60–72, Berlin, Germany, 1998. Springer Verlag.

[12] B. Bullnheimer, R.F. Hartl, and C. Strauss. A new rank based version of the ant system – a computational study. Technical report, Department of Management Science, University of Viena, 1997.

[13] B. Bullnheimer, R.F. Hartl, and C. Strauss. An improved ant system algorithm for thevehicle routing problem. *Annals of Operations Research*, 89: 319–328, January 1999.

[14] W. Burghout. *Hybrid microscopic-mesoscopic traffic simulation*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2004.

[15] O. Cordón, I.F. de Viana, and F.T. Herrera. Analysis of the best-worst ant system and its variants on the TSP. *Mathware & Soft Computation*, 9: 177–192, 2002.

[16] O. Cordon, I.F. de Viana, F.T. Herrera, and L. Moreno. A new ACO model integrating evolutionary computation concepts: The best-worst ant system. In M. et al. Dorigo, editor, *Abstract Proceedings of ANTS 2000- From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, pages 22–29, Belgium, 2000. IRIDIA, Universite Libre de Bruxelles.

[17] O. Cordon, F.T. Herrera, and T. Stützle. A review on the Ant Colony Optimization Metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9: 141–175, 2002.

[18] T. H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[19] T.J. Cova and J.P. Johnson. A network flow model for lane-based evacuation routing. *Transportation Research Part A*, 37: 579–604, 2003.

[20] J.L. Deneubourg, S. Aron, and J. Pasteels. The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, 3(2): 159–168, March 1990.

[21] G. Di Caro. *Ant Colony Optimization and its Application to adaptive Routing in Telecommunication Networks*. PhD thesis, Universite Libre de Bruxelles, Faculte de Sciences Appliquees, Brussels, Belgium, September 2004.

[22] G. Di Caro and M. Dorigo. AntNet: A mobile agents approach to adaptive routing. Technical Report 97-12, IRIDIA, Universite Libre de Bruxelles, Brussels, Belgium, 1997.

[23] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9: 317–365, 1998.

[24] M. Dorigo, M. Birattari, and T. Stützle. Ant colony optimization: Artificial ants as a computational intelligence technique. Technical report, IRIDIA-Technical Report Series, Universite Libre de Bruxelles, Bruxelles, Belgium, September 2006.

[25] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2): 137–172, 1999.

[26] M. Dorigo and L.M. Gambardella. A Study Of Some Properties Of Ant-Q. In I. Rechenberg H. M. Voigt, W. Ebeling and H. S. Schwefel, editors, *Proceedings of PPSN IV-Fourth International Conference on Parallel Problem Solving From Nature*, volume 1141, pages 656–665, Berlin, 1996. Springer-Verlag.

[27] M. Dorigo and L.M. Gambardella. Ant colonies for the Travelling Salesman Problem. *BioSystems*, 43: 73–81, 1997.

[28] M. Dorigo and L.M. Gambardella. Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem. *IEEE Transactions of Evolutionary Computation*, 1: 53–66, 1997.

[29] M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: An Autocatalytic Optimizing Process. Technical Report 91-016, Politecnico di Milano, Milano, Italy, 1991.

[30] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica-Politecnico di Milano, June 1991.

[31] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26: 29–41, 1996.

[32] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, July 2004.

[33] Marco Dorigo. *Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (Optimization, Learning, and Natural Algorithms)*. PhD thesis, Politecnico di Milano, Italy, 1992.

[34] D. R. Drew. *Traffic Flow Theory and Control*. McGraw-Hill, 1968.

[35] G. Eggenkamp. Dynamic multimodal route planning: an artificial intelligence approach. Master's thesis, Delft University of Technology, Faculty of Information Technology and Systems, July 2001.

[36] C.J. Eyckelhof and M. Snoek. Ant Systems for a Dynamic TSP. In M. Dorigo, G. Di Caro, and M. Samples, editors, *Ants Algorithms-Proceedings of ANTS 2002, Third International workshop on Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 88–99, London, UK, 2002. Springer-Verlag.

[37] M. Farooq and G. A. Di Caro. *Swarm Intelligence*, chapter Routing Protocols for Next-Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview, pages 101–160. Computer Science. Springer Berlin Heidelberg, 2008.

[38] D. B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscatawa, NJ, 1995.

[39] L. M. Gambardella and M. Dorigo. Ant-Q: A Reinforcement Learning approach to the Traveling Salesman Problem. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 252–260, 1995.

[40] B. George and S. Shekhar. *Advances in Conceptual Modeling - Theory and Practice*, volume 4231/2006, chapter Time-Aggregated Graphs for Modeling Spatio-temporal Networks, pages 85–99. Springer Berlin/Heidelberg, 2006.

[41] M. Guntsch and M. Middendorf. Pheromone modification strategies for ant algorithms applied to dynamic TSP. In *Applications of Evoluonary Computing: Proceedings of EvoWorkshops 2001*. Springer Verlag, 2001.

[42] M. Heusse, D. Snyers, S. Guérin, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. In *Advances in Complex Systems, Proceedings of ANTS'98, 1st International Workshop on Ant Colony Optimization*, pages 15–16, Brussels, Belgium, October 1998.

[43] S. P. Hoogendoorn and P. H. L. Bovy. State-of-the-art of vehicular traffic flow modelling. In *Journal of Systems and Control Engineering-Proceedings of the Institution of Mechanical Engineers*, volume Special Issue on Road Traffic Modelling and Control, pages 283–303, 2000.

[44] I. Kassabalidis, M.A. El-Sharkawi, R.J. II Marks, P. Arabshahi, and A.A. Gray. Swarm intelligence for routing in communication networks. In *Proceedings of the IEEE World Congress on Computational Intelligence*, Hawaii, May 2002.

[45] K. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *Proc. 10th Annual European Symposium on Algorithms*, pages 599–611. Springer, 2002.

[46] R. Kroon. Dynamic vehicle rouringusing ant based control. Master's thesis, Delft University of Technlogy, May 2002.

[47] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4): 408–416, 2009.

[48] A. M. Lazarescu. Entwicklung einer alternativen darstellung der netzebene für das verteilte verkehrsrouting in beejama. Master's thesis, Technische Universitat Dortmund, Fakultat fur Informatiek, April 2009.

[49] R. Li. A simulation system for hierarchical routing using ant based control. Master's thesis, Delft University of Technology, July 2004.

[50] S. Liang, A. N. Zincir-Heywood, and M. I. Heywood. The effect of routing under local information using a social insect metaphor. In *CEC '02: Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress*, pages 1438–1443, Washington, DC, USA, May 2002. IEEE Computer Society.

[51] S. Liang, A. N. Zincir-Heywood, and M. I. Heywood. Intelligent packets for dynamic network routing using distributed genetic algorithm. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 88–96, San Francisco, CA, USA, July 2002. Morgan Kaufmann Publishers Inc.

[52] V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the Quadratic Assignment Problem. *INFORMS Journal on Computing*, 11(4): 358–369, 1999.

[53] D. Metz. Predictive navigation: the case for an initiative. Technical report, Centre for Transport Studies, University College London, 2009.

[54] M. Mirshahi, J. Obenberger, C. A. Fuhs, C. E. Howard, R. A. Krammes, B. T. T. Kuhn, R. M. Mayhew, M. A. Moore, Stone C. J. Sahebjam, K., and J. L. Yung. Active traffic management: The next step in congestion management. Technical report, U.S. Department of Transportation, American Association of State Highway, July 2007.

[55] Ouintiq. Quintiq version 4.2.5 Documentation. Technical report.

[56] E. J. Schmitt and H. Jula. Vehicle Route Guidance Systems: Classification and Comparison. pages 242 –247, September 2006.

[57] R. Schoonderwoerd, O. Holland, and J. Bruten. Ant-like agents for load balancing in telecommunications networks. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 209–216, New York, USA, 1997. ACM.

[58] R. Schoonderwoerd, O.E. Holland, J. Bruten, and L. J. M. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2): 169–207, 1996.

[59] R. Sondak. Dynamic Intelligent Algorithm for Navigation. Master's thesis, Delft University of Technology, In press.

[60] T. Stützle. *Local Search Algorithms for Combinatorial Problems Analysis, Improvements, and New Applications*. PhD thesis, Technischen UniversitÍat Darmstadt, 1998.

[61] T. Stützle and H.H. Hoos. Improving the Ant System: A Detailed Report on the MAX-MIN Ant System. Technical Report AIDA-96-12, FB Informatiek, TU Darmstadt, August 1996.

[62] T. Stützle and H.H. Hoos. MAX-MIN Ant System. *Future Gener. Comput. Syst.*, 16(9): 889–914, June 2000.

[63] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of International Joint Conference on Artifical Intelligence 1997*, volume 2, pages 832–838, Palo Alto, CA, 1998. Morgan Kauffman.

[64] B. Tatomir, H. Dibowski, and L. J. M. Rothkrantz. Hierarchical routing in traffic networks. In *Proceedings of 16th Belgian-Dutch Conference on Artificial Intelligence (BNAIC 2004)*, pages 75–82, October 2004.

[65] B. Tatomir and L. J. M. Rothkrantz. H-ABC: A scalable dynamic routing algorithm. In *Recent Advances in Artificial Life*, number 3, pages 279–293. World Scientific Publishing Co. Pte. Ltd., Singapore, November 2005.

[66] B. Tatomir and L. J. M. Rothkrantz. Hierarchical routing in traffic using swarm-intelligence. In *Proceedings of The 9th International IEEE Conference on Intelligent Transportation Systems 2006*, pages 230–235, Toronto, Canada, September 2006.

[67] Tom Tom. Whitepaper, How TomTom's HD Traffic and IQ Routes data provides the very best routing. Technical report, Tom Tom BV.

[68] H. Van Lint, H. Van Zuylen, S. Hoogendoorn, A. Hegy, M. Bliemer, and A. Pel. *Traffic Theory for Intelligent Transport Systems for Road Transport.*

[69] J. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers, Part II*, 1(36): 352–362, 1952.

[70] H. F. Wedde, S. Lehnhoff, B. van Bonn, Z. Bay, S. Becker, S. Boettcher, C. Brunner, A. Buescher, T. Fuerst, A.M. Lazarescu, E. Rotaru, S. Senge, B. Steinbach, F. Yilmaz, and T. Zimmermann. Highly dynamic and adaptive traffic congestion avoidance in real-time inspired by honey bee behavior. In *PEARL 2007 - Informatik Aktuell - Mobilitaet und Echtzeit*, pages 21–31. Springer, Boppard, Germany, 2008.

[71] H.F. Wedde, M. Farooq, and Y. Zhang. BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior. In *Lecture Notes in Computer Science*, volume 3172 of *Lecture notes in computer science*, pages 83–94. Springer, Berlin, 2004.

[72] H.F. Wedde, S. Lehnhoff, B. van Bonn, Z. Bay, S. Becker, S. Bottcher, C. Brunner, A. Buscher, T. Furst, A.M. Lazarescu, E. Rotaru, S. Senge, B. Steinbach, F. Yilmaz, and T. Zimmermann. A novel class of multi-agent algorithms for highly dynamic transport planning inspired by honey bee behavior. In *ETFA'07 - Proceedings of the 12th conference on emerging technologies and factory automation*, pages 1157 – 1164, Patras, September 2007. IEEE Press.

[73] S. Winter. Modeling costs of turns in route planning. *GeoInformatica*, 6: 345–361, December 2002.

[74] Karl E. Wunderlich, David E. Kaufman, and Robert L. Smith. Link travel time prediction for decentralized route guidance architectures. *IEEE Transactions on Intelligent Transportation Systems*, 1(1):4–14, 2000.

[75] H. Xu. *Decentralized Traffic Information System Design Based on Inter-Vehicle Communication.* PhD thesis, University of California Riverside, December 2006.

Appendices

Appendix A

Class Diagram

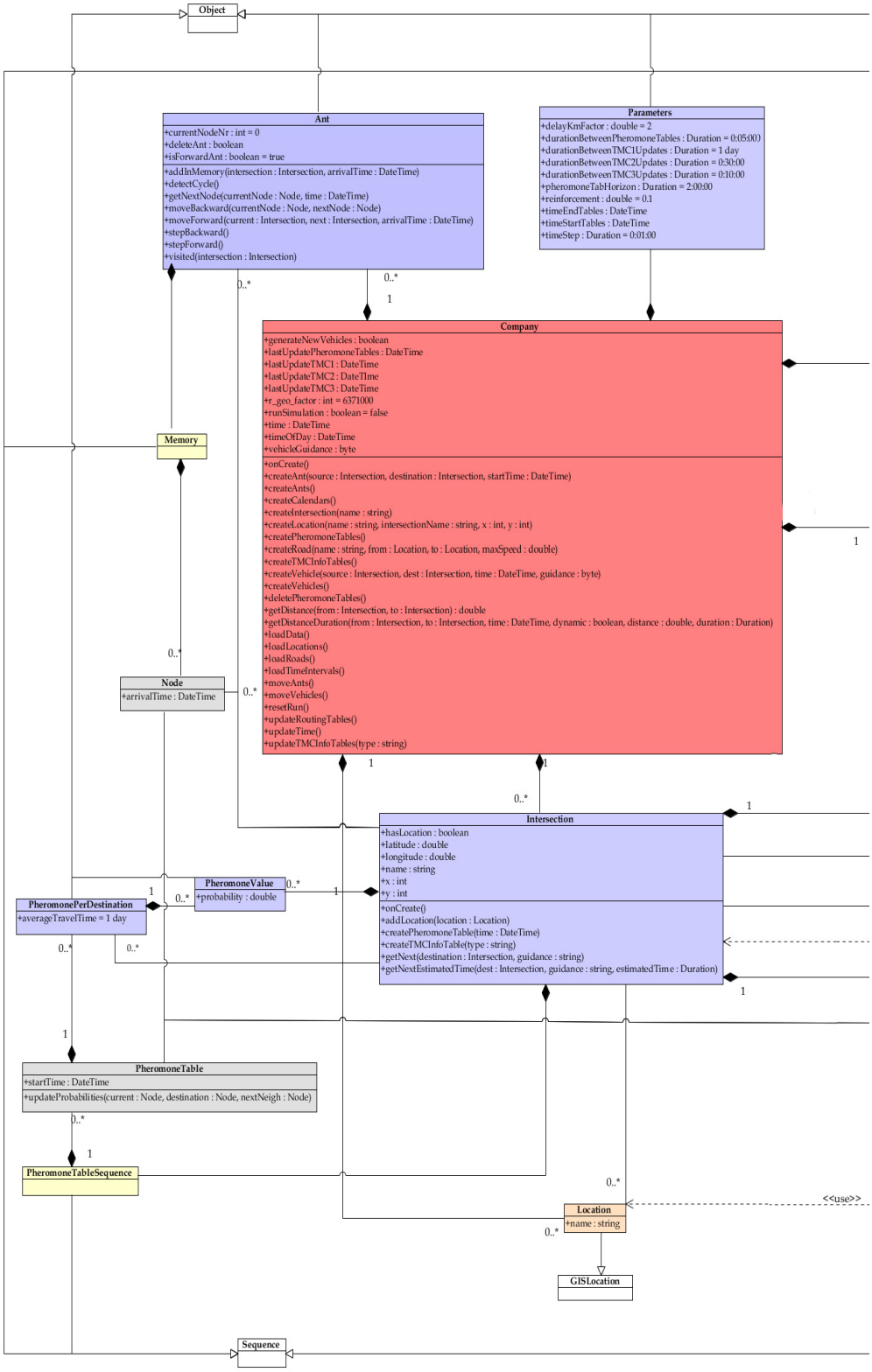


Figure A.1: Complete Class Diagram. Part A.

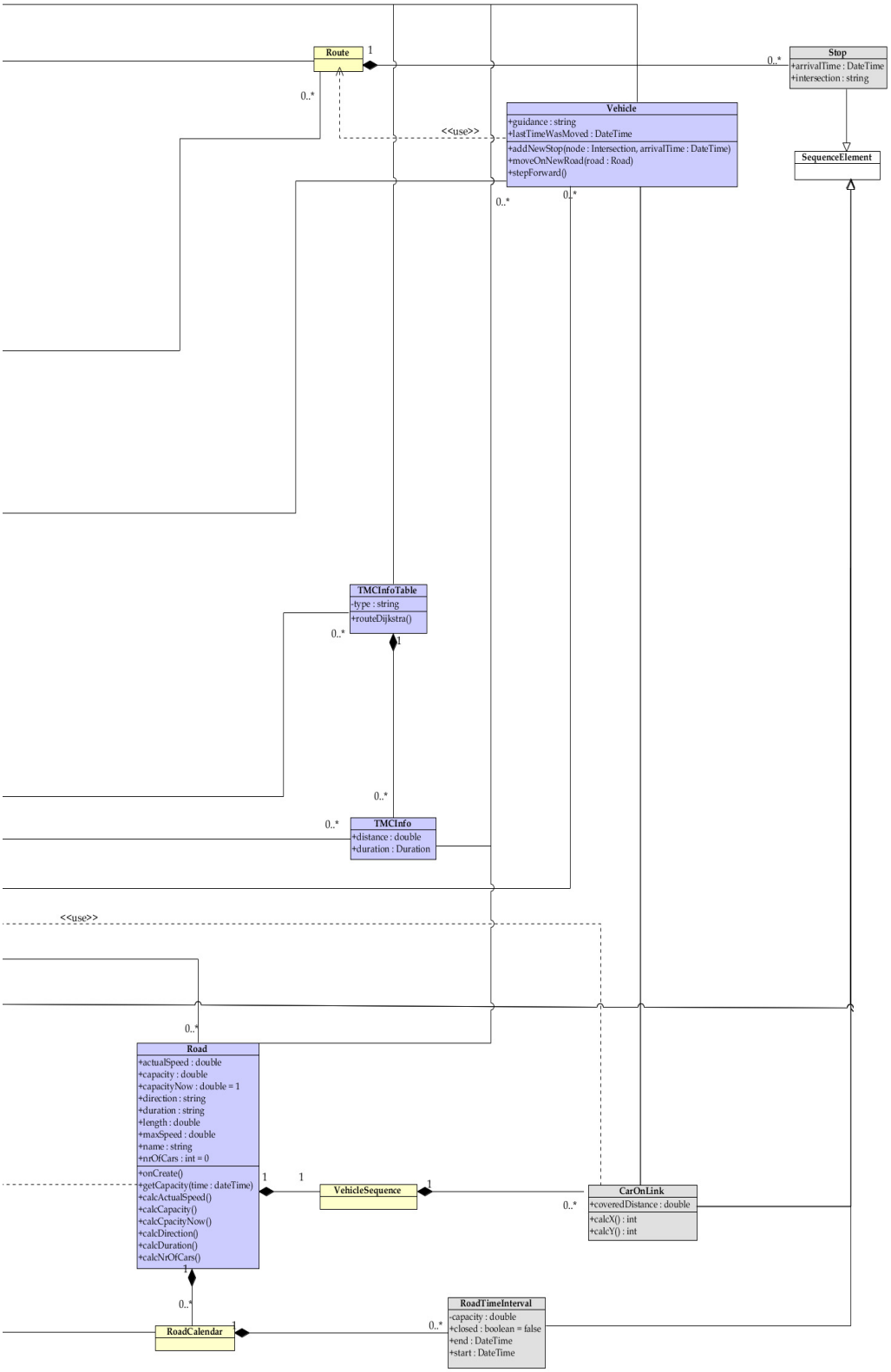


Figure A.2: Complete Class Diagram. Part B.

Appendix B

Results of the adaptivity test

Table B.1: Distribution of routes selected in percentages.

Percentages reflect how many times a route is selected by ABC, static Dijkstra, DD 10 min and DD 30 min when leaving from the source Hintham to destination Ypenburg. Percentages are calculated in the context of the traffic situation from 27.09.2007.

Route	ABC	Static Dijkstra	DD 10 min	DD 30 min
Route 1	13.56 %	0 %	100 %	93.44 %
Route 2	37.29 %	100 %	0 %	6.56 %
Route 3	5.08 %	0 %	0 %	0 %
Route 4	11.86 %	0 %	0 %	0 %
Route 5	30.51 %	0 %	0 %	0 %
Route 6	1.69 %	0 %	0 %	0 %

Table B.2: Paths selected by ABC from Hintham to Ypenburg.

Segment	**Route 1**	**Route 2**	**Route 3**	**Route 4**	**Route 5**	**Route 6**
	Hintham	Hintham	Hintham	Hintham	Hintham	Hintham
1	Empel	Empel	Empel	Empel	Empel	Empel
2	Deil	Deil	Deil	Hooipolder	Hooipolder	Hooipolder
3	Everdingen	Gorinchem	Gorinchem	Gorinchem	Zonzeel	Zonzeel
4	Oudenrijn	Ridderkerk	Ridderkerk	Ridderkerk	Klaverpolder	Klaverpolder
5	Gouwe	Terbregseplein	Vaanplein	Terbregseplein	Ridderkerk	Ridderkerk
6	Pr. Clausplein	Kleinpolderplein	Beneluxplein	Kleinpolderplein	Terbregseplein	Vaanplein
7	Ypenburg	Ypenburg	Kethelplein	Ypenburg	Kleinpolderplein	Beneluxplein
8			Kleinpolderplein		Ypenburg	Kethelplein
9			Ypenburg			Kleinpolderplein
10						Ypenburg
Length	100,6 Km	99,9 Km	110 Km	112 Km	100,1 Km	116,1 Km
Segments	7	7	9	7	8	10

Table B.3: The path selected by Static Dijkstra from Hintham to Ypenburg.

Segment		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Length**
Route 2	Hintham	Empel	Deil	Gorinchem	Ridderkerk	Terbregseplein	Kleinpolderplein	Ypenburg	99,9 Km

Table B.4: The path selected by DD 30 min from Hintham to Ypenburg.

Segment		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Length**
Route 1	Hintham	Empel	Deil	Everdingen	Oudenrijn	Gouwe	Prins Clausplein	Ypenburg	100,6 Km

Table B.5: The paths selected by DD 10 min from Hintham to Ypenburg.

Segment		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Length**
Route 1	Hintham	Empel	Deil	Everdingen	Oudenrijn	Gouwe	Prins Clausplein	Ypenburg	100,6 Km
Route 2	Hintham	Empel	Deil	Gorinchem	Ridderkerk	Terbregseplein	Kleinpolderplein	Ypenburg	99,9 Km

Table B.6: Distribution of the routes selected by ABC, 7:00-12:00 27.09.2007.

Time	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6
7:00	-	x	-	-	-	-
7:05	x	-	-	-	-	-
7:10	-	-	-	-	x	-
7:15	x	-	-	-	-	-
7:20	x	-	-	-	-	-
7:25	-	x	-	-	-	-
7:30	-	-	x	-	-	-
7:35	-	-	x	-	-	-
7:40	-	-	-	x	-	-
7:45	-	-	x	-	-	-
7:50	-	x	-	-	-	-
7:55	-	x	-	-	-	-
8:00	-	-	-	-	x	-
8:05	-	-	-	-	x	-
8:10	-	-	-	-	x	-
8:15	-	x	-	-	-	-
8:20	-	x	-	-	-	-
8:25	-	-	-	x	-	-
8:30	-	-	-	-	x	-
8:35	-	x	-	-	-	-
8:40	-	x	-	-	-	-
8:45	-	x	-	-	-	-
8:50	-	-	-	-	-	x
8:55	-	x	-	-	-	-
9:00	-	-	-	-	x	-
9:05	-	-	-	-	x	-
9:10	-	-	-	x	-	-
9:15	-	-	-	-	x	-
9:20	-	-	-	x	-	-
9:25	-	-	-	-	x	-
9:30	-	-	-	-	x	-
9:35	-	x	-	-	-	-
9:40	-	-	-	-	x	-
9:45	-	x	-	-	-	-
9:50	-	-	-	-	x	-
9:55	-	-	-	-	x	-
10:00	-	x	-	-	-	-
10:05	-	x	-	-	-	-
10:10	-	-	-	x	-	-

Time	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6
10:15	-	-	-	-	x	-
10:20	-	-	-	-	x	-
10:25	-	-	-	-	x	-
10:30	-	-	-	-	x	-
10:35	-	-	-	x	-	-
10:40	-	x	-	-	-	-
10:45	-	x	-	-	-	-
10:50	-	x	-	-	-	-
10:55	-	x	-	-	-	-
11:00	-	x	-	-	-	-
11:05	x	-	-	-	-	-
11:10	-	x	-	-	-	-
11:15	-	x	-	-	-	-
11:20	-	x	-	-	-	-
11:25	-	-	-	x	-	-
11:30	x	-	-	-	-	-
11:35	-	-	-	-	x	-
11:40	-	-	-	x	-	-
11:45	-	-	-	x	-	-
11:50	x	-	-	-	-	-
11:55	x	-	-	-	-	-
12:00	x	-	-	-	-	-

Appendix C

Paper from *Proceedings of ANTS 2008*

Dynamic Routing and Travel Time Prediction with Ant Based Control[†]

Bogdan Tatomir[1], Adriana-Camelia Suson[2], and Leon Rothkrantz[2]

[1]Quintiq,'s Hertogenbosch, The Netherlands
bogdan.tatomir@quintiq.com
[2]Delft University of Technology, Delft, The Netherlands
{AC.Suson,L.J.M.Rothkrantz}@ewi.tudelft.nl

At this moment the capacity of the highways is not sufficient to transport all car drivers without delay. Especially in the rush hours there are enormous traffic jams. In case of special events such as traffic accidents car drivers can be delayed by hours. The losses in time and money are enormous, so the problem of traffic congestion has a high priority.

To increase the capacity of the road network is not an option. The construction of new freeways in some areas is blocked by ecological, financial or political reasons. On the short term the only solution is to use the road network in an optimal way. Usually, there are many alternatives to travel from A to B. In case one route is blocked or delayed car drivers should be informed about alternatives routes. Nowadays, information about traffic jams is broadcasted via news on radio and TV, or via mobile phones, but normally only big congestions are reported and no alternative routes are suggested. Most current route planner devices have TMC (Traffic Message Channel) integrated but their information is updated only every 30 minutes which is not so fast. For extra payment special services provide personalized information which gets updated every few minutes. But all these services use the situation of the roads on a given moment and are unable to take the future into account. If all the cars are advised to take the same alternative route, soon also this one will get congested.

To compute alternatives routes it is necessary to have good prediction models of expected congestions and fast algorithm to compute the shortest path while being able to react to dynamic changes in the network caused by special incidents. In this paper we present a dynamic routing system founded on Ant Based Control (ABC). Starting from historical traffic data, ants are used to compute and predict the travel times along

[†]*Published in Proceedings of the 6th International Conference on Ant Colony Optimization and Swarm Intelligence. 2008.*

the road segments. They are finding the fastest routes not only looking to the past and present traffic conditions but also trying to anticipate and avoid future congestions.

Various shortest path algorithms are available for computing the optimal route. The most popular algorithm is Dijkstra's algorithm that has a runtime complexity of $O(n^2)$, where n is the number of nodes in the network. Many variations to the Dijkstra's algorithm such as bidirectional search and binary heap implementation have been proposed to improve its response time. An improved version of the Dijkstra's algorithm is the A^* algorithm [1], which is widely used in vehicle navigation.

Static routing algorithms like Dijkstra's algorithm only apply to central routing. To minimize the traveling time we need a decentralized routing algorithm which is able to adapt to the dynamic changes that take place in the traffic network. We found the answer in the real life behaviour of ants. Despite they are very simple insects, together, in a colony, they show what is called emergent behaviour and are able to accomplish complex tasks. Using the laying of pheromone ants are able to find the shortest path from their nest to a food source and vice-versa.

In [2] a dynamic vehicle routing system was introduced which uses the Ant Based Control algorithm (ABC-algorithm) for car navigation in a city. But the algorithm proved to be suitable for small networks and showed scalability problems on big networks as the one of the streets of a city. This problem was solved in [3] where the H-ABC, a scalable ant colony optimization algorithm was presented. Similar with the current navigation systems the previous algorithms looks only at the past and present and no future is considered.

To predict travel time, historical data (recorded from the ANWB) can be used, especially when the prediction horizon becomes further away. Apparently the load on the freeway network is almost the same on compatible days (same day of the week). Unfortunately, this has one considerable drawback: when unexpected events (like accidents) happen, a travel time prediction based on historical data will be completely wrong. Our approach is to use an Ant Based Control algorithm to approximate how many cars are expected to travel between two nodes in a specific time interval. Knowing the traffic flow, based on the speed/density relation of the traffic, we can make an estimate of the expected travelling time on a road segment.

To test our concepts we modeled a part of the Dutch highway network. It consists of 58 nodes (highway intersections) and 84 bidirectional roads characterized by the number of lanes and the maximum allowed speed. Because of the big amount of data, we focused only on the morning period between 5:00 and 12:00.

For almost 57% of the 3306 possible routes, faster alternatives were found by our routing system compared with a static one using with the Dijkstra's algorithm. A difference from 10 to 20 minutes was noticed in 12.16% of the situations. If we consider that The Netherlands is a small country and most of the selected routes are shorter than 150 km, using the dynamic routing system is a real benefit.

Bibliography

[1] I.Chiabini and S. Lan. Adaptations of the A* Algorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Networks. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):60-74, 2002.

[2] B.Tatomir and L.J.M.Rothkrantz. Dynamic traffic routing using Ant Based Control. *International Conference on Systems, Man and Cybernetics IEEE SMC*, pages 3970-3975, 2004.

[3] B. Tatomir and L.J.M. Rothkrantz. Hierarchical routing in traffic using swarm intelligence. In *Proceedings of the 9th International IEEE Conference on Intelligent Transportation systems 2006*, pages 230-235, Toronto, Canada, September 2006.

Appendix D

Paper from *Proceedings of the 2009 Winter Simulation Conference*

TRAVEL TIME PREDICTION FOR DYNAMIC ROUTING USING ANT BASED CONTROL

Bogdan Tatomir
Leon J.M. Rothkrantz
Adriana C. Suson

Delft University of Technology
Mekelweg 4, Delft, 2628 CD
THE NETHERLANDS

ABSTRACT

Currently most car drivers use static routing devices based on the shortest distance between start and end position. But the shortest route can differ from the shortest route in time. To compute alternative routes it is necessary to have good prediction models of expected congestions and a fast algorithm to compute the shortest path while being able to react to dynamic changes in the network caused by special incidents. In this paper we present a dynamic routing system based on Ant Based Control (ABC). Starting from historical traffic data, ants are used to compute and predict the travel times along the road segments. They are finding the fastest routes not only looking to the past and present traffic conditions but also trying to anticipate and avoid future congestions.

1 INTRODUCTION

At this moment the capacity of the highways is not sufficient to transport all car drivers without delay. Especially in the rush hours there are enormous traffic jams. In case of special events such as traffic accidents car drivers can be delayed by hours. The losses in time and money are enormous, so the problem of traffic congestion has a high priority.

To increase the capacity of the road network is not an option. The construction of new freeways in some areas is blocked by ecological, financial or political reasons. On the short term the only solution is to use the road network in an optimal way. Usually, there are many alternatives to travel from A to B. In case one route is blocked or delayed car drivers should be informed about alternative routes. Nowadays, information about traffic jams is broadcasted via news on radio and TV, or via mobile phones, but normally only big congestions are reported and no alternative routes are suggested. Most current route planner devices have TMC (Traffic Message Channel) integrated but their information is updated only every 30 minutes which is not so fast. For extra payment special services provide personalized information which gets

[†]Published in the *Proceedings of the 2009 Winter Simulation Conference. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.*

updated every few minutes. But all these services use the situation of the roads at a given moment and are unable to take the future into account. If all the cars are advised to take the same alternative route, this one will also get congested.

To efficiently route car drivers, besides a detailed information of the road network, it is necessary to know the travel speed along road segments as a function of time. In many countries travel speed on the motorways is measured by special devices as: cameras, wires in the road and tracking car devices. Combining this data with historical data, travel time predictions can be computed. In this project we have access to the data of ANWB, an organization involved in traffic management.

In this paper we describe a system for providing dynamic routing information to car drivers. It uses Ant Based Control algorithm which is able, based on current and historical traffic information, to avoid congestions by predicting the future load on the roads. The outline of the paper is as follows. In the next section we present and discuss related work. The following section contains a description of the data and of the model used by our approach for time prediction. In Section 4 is presented the new ABC algorithm we use for dynamic routing. The experiments and results are described in Section 5. We end the paper with conclusions and future work.

2 RELATED WORK

Various shortest path algorithms are available for computing the optimal route. The most popular algorithm is Dijkstra's algorithm that has a runtime complexity of $O(n^2)$, where n is the number of nodes in the network. Many variations to the Dijkstra's algorithm such as bidirectional search and binary heap implementation have been proposed to improve its response time. The A^* algorithm (Chabini and Lan 2002), which is widely used in vehicle navigation, is an improved version of the Dijkstra's algorithm. It makes use of an appropriate heuristic function to search the most promising nodes first thereby reducing the computation time.

Static routing algorithms like Dijkstra's algorithm only apply to central routing. To minimize the traveling time we need a decentralized routing algorithm which is able to adapt to the dynamic changes that take place in the traffic network. We found the answer in the real life behaviour of ants. Despite they are very simple insects, together, in a colony, they show what is called emergent behaviour and are able to accomplish complex tasks. They lay a special sort of volatile hormone, or pheromone, to create a signalling system between them. Using pheromone trails ants are able to find the shortest path from their nest to a food source and vice-versa.

Ant-based algorithms have successfully been applied for dynamic routing in different types of problems. First they were used for packet routing in networks ((Schoonderwoerd, Holland, Brunten and L.J.M.Rothkrantz 1996) and (Di Caro and Dorigo 1998)) and in wireless networks ((Gunes, Sorges, and Bonazizi 2002) and (Di Caro, Ducatelle, and Gambardella 2005)). Recently, ant colony optimization was applied to the vehicle routing problems with time-dependent travel times (Ichoua, Gendreau, and Potvin 2003) and (Donati, Montemanni, Casagrande, Rizzoli, and Gambardella 2008).

In (Tatomir and Rothkrantz 2004) a dynamic vehicle routing system was introduced which uses the Ant Based Control algorithm (ABC-algorithm) for car navigation in a city. But the algorithm proved to be suitable for small networks and showed scalability problems on big networks as the one of the streets of a city. This problem was solved in (Tatomir and Rothkrantz 2006) where the H-ABC, a scalable ant colony optimization algorithm was presented. Similar to current navigation systems the previous algorithms look only at the past and present and no future is considered. Some ideas of using ABC for travel time prediction are presented in (Ando, Masutani, Sasaki, Iwasaki, Fukazawa, and Honiden 2005) and (Weyns, Holvoet, and Helleboogh 2007).

Many other different traffic prediction approaches can be found in the recent literature. Jagadeesh et al. (Jagadeesh, Srikanthan, and Quek 2002) present an efficient hierarchical routing algorithm that finds a near-optimal route and evaluate it on a large city road network. In (Rice and van Zwet 2004) is described a method to predict the time that will be needed to traverse a given section of a freeway when the departure is at a given time in the future. The prediction is done on the basis of the current traffic situation in combination with historical data. In (Wu, Ho, and Lee 2004) support vector regression (SVR) is applied for travel-time prediction. The authors of (Kim, Lewis, and White 2005a) developed a decision-making procedures for determining the optimal driver attendance time, optimal departure times and optimal routing policies under time-varying traffic flows based on a Markov decision process formulation. The same authors in (Kim, Lewis, and White 2005b) model the dynamic route determination problem as a Markov decision process (MDP) and present procedures for identifying traffic data having

no decision-making value.

3 TRAVEL TIME MODELING

We consider a traffic network of highways composed of roads segments and intersections (see Figure D.1). The most straightforward approach to model the problem would be to construct a graph. As with other routing problems, Dijkstra's shortest path algorithm could be used to find the shortest path between origin and destination in this graph. However, in the case of dynamic route planning, representing the problem through a graph $G = (N, E)$ is not as straightforward as with a 'normal' routing problem, when the nodes $(N = 1, 2, \ldots, n)$ represent the junctions and the links $(E = (e_j)_{j \leq m})$ represent the roads between them with fixed travel times.

When trying to represent the dynamic route planning problem, two major issues have to be dealt with. Firstly, the dynamic aspect of the data has to be taken into account. The travel time between the different junctions changes in time, and somehow these changes have to be taken into account and incorporated in the graph. When, for example, travelling from Amsterdam to Delft in the morning rush hour, a departure with only 5 minutes later, can affect the travel time by more than 20 minutes, since major congestion may have occurred along the route during these 5 minutes. For example, an accident might have happened or a sudden peak in cars that want to access the highway may have occurred.

To model the dynamic travel time t_{ij} along the road segments we add vertical time axis and discretize the time into intervals $(I_0, I_1, \ldots, I_n)$, where $I_k = [start_k, end_k]$. At each time intervals we make a parallel copy of the (x,y) plane but with different travel times $t_{ij}(I_k)$.

An example of a network constructed this way is shown in Figure D.2. The original graph consisting of nodes A to G is repeated for each time interval. The edges between the nodes A to G (the lowest graph at I_0) represent which nodes are connected to each other. For clarity these edges have been kept in the different layers of the graph to show these connections. The dotted links that intersect the different layers are the actual road connections. For each layer and for all nodes, outgoing links are constructed according to the travel time at that moment. It should be noticed that in Figure D.2 not all the links are shown, since that would have resulted in a cluttered figure.

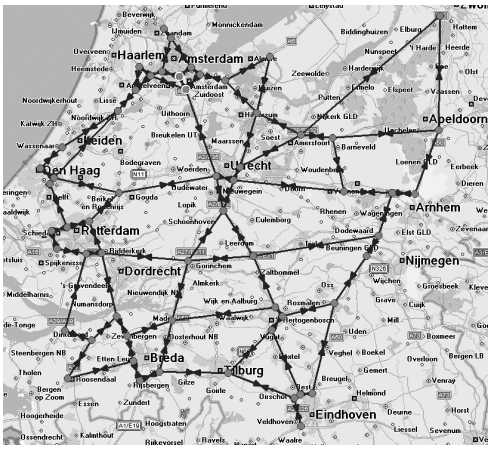


Figure D.1: Map of important highways in the Netherlands.

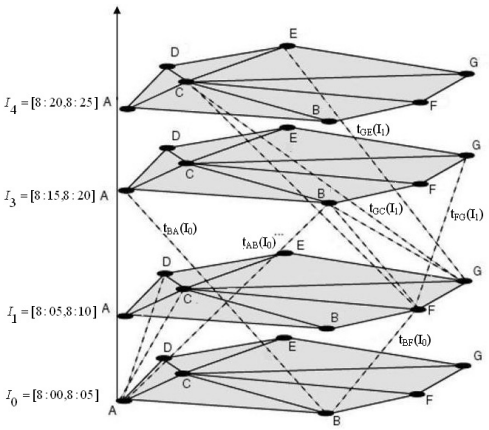


Figure D.2: A 3D time dependent model of a road network

3.1 Travel Time Prediction

To predict travel time, historical data can be used, especially when the prediction horizon is moved further away in time. Apparently the load on the freeway network is almost the same on compatible days (same day of the week). We used historical data recorded from the ANWB. In Figure D.3 you can see how the actual travelling speed on a road evolved in one morning. Unfortunately, this approach has one considerable drawback: when unexpected events, like

accidents, happen a travel time prediction based on historical data will be completely wrong. Somehow, the actual state of the network has to be taken into account, especially when predicting travel times in the near feature. In the following section an Ant Based Control algorithm is used to approximate the travelling speed between two nodes. It will try to predict how many cars are expected to travel between nodes i and j in the time interval I_k. Based on the speed/density relation of the traffic (Figure D.4) we can make an estimate of the expected travelling time on that segment during I_k.

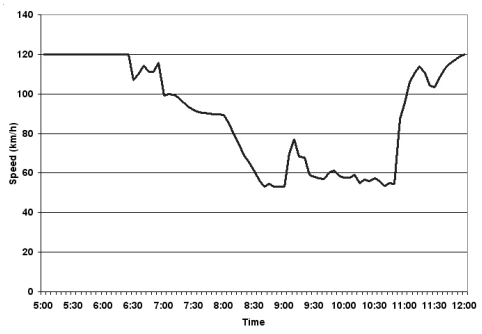


Figure D.3: Travel speed between Deil and Tiel as a function of time

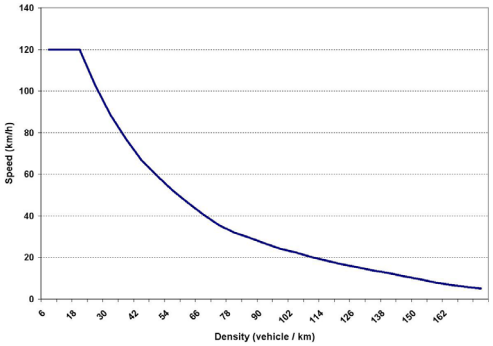


Figure D.4: Traffic speed/density relation

We use the following formula to compute the average speed when travelling between A and B in the time interval I_k:

$$v_{AB}(I_k) = \tau h_{AB}(I_k) + (1-\tau) f_{AB}(I_k) \tag{D.1}$$

where $h_{AB}(I_k)$ is the travel speed based on historical data in the interval I_k and $f_{AB}(I_k)$ is the predicted travel speed. We chose τ=0.5 to balance the past information with the future prediction.

Next we present the algorithm used to compute the travelling time $t_{ij}(I_k)$ between i and j when starting in interval I_k.

3.1.1 *GetTravelTime*(i, j, t_i)
{n - the time intervals horizon}
{i - current node, j - successor node of i}
{t_i - departure time from node i}
$L \leftarrow d(i,j)$ {$d(i,j)$ - distance between i and j}
$t \leftarrow t_i$
$k \leftarrow 0$ {start the iteration with the first time interval of that road}
while $k \leq n$ and $L > 0$ **do**
  if $t \in I_k$ **then**
    $v_{ij}(I_k) \leftarrow \tau(h_{ij}(I_k))+(1-\tau)(f_{ij}(I_k))$ {compute the average speed} {check if the remained distance can be traversed in this time interval}
    if $L \leq v_{ij}(I_k)(I_k.end - t)$ **then**
      $t \leftarrow t + \frac{L}{v_{ij}(I_k)}$
      $L \leftarrow 0$
    else
      $L \leftarrow L - v_{ij}(I_k)(I_k.end - t)$
      $t \leftarrow I_k.end$
    end if
  end if
  $k \leftarrow k + 1$
end while
return $t - t_i$

4 PREDICTIVE ANT BASED CONTROL ALGORITHM

4.1 Predictive Model

We developed a variant of ABC to predict travel speeds. In the classical ABC, the ants were travelling between random generated pairs of source and destination. In our case most of the ants will be created because of the presence of a car on the roads. They will have as source the starting node of the car or, a more probable situation, the next highway intersection the car is driving to. The destination will be the same one as for the related car. Some of the ants which travel between random nodes. This is useful especially at the beginning of the simulation for training the routing tables. Also in case the traffic flow is very low, we want to maintain the routing tables up to date.

The mentioned routing tables are also modified. Every node will keep now multiple layers of probability tables, one for each time interval. $P_{dn}(I_k)$ represents the probability to go to destination d via neighbour n when you are in node i at time $t \in I_k$. Besides probability tables every node i has the following structures:

- $\mu_d(I_k)$: stores the average travel time from node i to destination d when starting at node i in the interval I_k.
- $\delta_n(I_k)$: stores the cars expected to go during I_k to next node n. It contains pairs (x, t_x) where x is the car id plate and t_x is the time stamp when the corresponding ant visited the node. Each time a new ant generated for car x comes to a node, it updates the value of t_x. A timer checks periodically if $\delta_n(I_k)$ is up-to-date and removes the old records (old t_x - no update was received for car x).
- $\rho_n(I_k)$ is the expected traffic density on the road (i, n) during I_k. It is based on the size of $\delta_n(I_k)$ and is used to compute the predicted travel speed $f_{in}(I_k)$.

We are now going to describe some of the features of the dynamic ABC algorithm. Its characteristics make it suitable for both central and distributed implementations.

4.2 Forward Ants

The forward ants behave similar with the forward ants in *AntNet* (Di Caro and Dorigo 1998). They keep a memory about the visited nodes and the estimated time to reach them. As mentioned before they are created periodically for each car in the network. Still, at every source node s in the network the random forward ants $F_{sd}(t_0)$ are generated. Not only their destination d is chosen but also the starting time t_0 is selected as the current time or close in the future (5-15 min away). In this way the ants not only determine the shortest paths for the present but they are training the routing tables in advance for the near future.

At every node i, the selection of the next node n, to move to, is done according to the probabilities P_d. If a cycle is detected ant is deleted.

Lets consider t_{si} the estimated arrival time at node i coming from source s. Once arrived at node i and before going to the next neighbour n, the ant calculates the estimated travelling time between i and n when the departure is in the interval I_k. t_{sn} represents the estimated time necessary for a car to travel from intersection s to intersection n.

$$t_{sn} \leftarrow t_{si} + GetTravelTime(i, n, t_{si}) \tag{D.2}$$

A forward ant reaches its destination when it arrives at node d.At this moment the ant $F_{sd}(t_{sd})$ finishes its trip. It transfers all of its memory to a new generated backward ant B_{ds} and dies.

4.3 Backward Ants

A backward ant takes the same path as that of its corresponding forward ant, but in the opposite direction. Its role is to update the routing tables along the path based on the travel time estimations collected by the the forward ant. At every step it pops its stack to know the next hop node.

Once arrived at node i the backward ant $B_{ds}(t_{si})$ first selects the time interval $t_{si} \in I_k$ for which the routing table should be updated. The updates are done for all the nodes j between i and d. This is done in 5 steps:

1. Get estimate traveling time from node i to node j when starting in the interval I_k at node i:

$$t_{ij} = t_{sj} - t_{si} \tag{D.3}$$

2. Update the average travelling time $\mu_j(I_k)$:

$$\mu_j(I_k) = \mu_j(I_k) + \eta(t_{ij} - \mu_j(I_k)) \tag{D.4}$$

   We used $\eta = 0.1$. It represents the impact a measurement will have in time over the average value.

3. Compute the reinforcement r to be used to update the routing table. This is a function of the time t_{ij} and its mean value $\mu_j(I_k)$.

$$r = \begin{cases} \frac{t_{ij}}{c\mu_j(I_k)};\ \frac{t_{ij}}{c\mu_j(I_k)} < 1,\ c = 1.1 > 1 \\ \\ r = 1 \text{ otherwhise} \end{cases} \tag{D.5}$$

4. Update the routing table for destination j:

$$P'_{jn}(I_k) = P_{jn}(I_k) + (1 - r)(1 - P_{jn}(I_k)) \text{ for the link } i{\rightarrow}n \tag{D.6}$$

$$\text{otherwise } P'_{jl}(I_k) = P_{jl}(I_k) - (1 - r)P_{jl}(I_k),\ \text{for } l \neq n \tag{D.7}$$

5. If $r < 1$ and F_{sd} was generated for a car x, update $\delta_n(I_k)$ with the pair (x, t_{now}). If is the first time the car x is expected in node i, a new record is created. Otherwise the old record (x, t_{old}) is replaced by the new one. Be aware that t_{old} and t_{now} are system times and not calculated by ants.

When the source node s is reached again, the ant B_{ds} is deleted.
Next we present the pseudo code of the algorithm.

4.3.1 ABC Algorithm

```
{i - current node, d - destination node, s - source node}
{n - successor node of i, p - predecessor node of i}
for all Nodes do
  if time to generate an ant at node s then
    for all now and next k time intervals do
      Create F_sd(t_k)
    end for
  end if
  for all forward ants F_sd(t_si) received at node i do
    if cycle detected then
      Remove F_sd(t_si)
    else
      if i = d {destination reached} then
        Create B_ds(t_sd)
        p ← GetPrev(F_sd(t_sd)) {select previous node}
        Move B_ds(t_sd) to p
        Remove F_sd(t_sd)
      else
        n ← GetNext(F_sd(t_si))
        t_sn ← t_si + GetTravelTime(i, n, t_si) {compute the travelling time}
        F_sd ← (n, t_sn) {add the new information on the stack}
        Move F_sd(t_sn) to n
      end if
    end if
  end for
  for all backward ants B_ds received at node i do
    UpdateRoutingTables(i, i ← d) {update the node information}
    if d ≠ i {destination not reached} then
      n ← GetNext(B_ds(t_si) {select next node to go}
```

Move $B_{ds}(t_{sn})$ to n
else
Remove $B_{ds}(t_{si})$
end if
end for
end for

4.4 Tests and Results

To test our concepts we modelled a part of the Dutch highway network (see Figure D.1). For this we used the Quintiq software which has an integrated GIS system. The network we built consists of 58 nodes (highway intersections) and 84 bidirectional roads. Each road is characterized by the number of lanes and the maximum allowed speed. Some of the real roads are composed of multiple segments with different speed limits and number of lanes. In this cases we choose an average speed limit and a fixed number of lanes. For historical data we used data collected from the ANWB website. Because of the big amount of data, we focused only on the morning period between 5:00 and 12:00. In all the nodes we created routing tables for every 10 min interval.

We populated the roads with random traffic which was generated between the intersections. We used four types of vehicles. 25% of them were guided using the ABC dynamic routing. 25% received traffic update every 30 minutes like the common car navigators are working. Then 25% of the cars received traffic update every 10 minutes like a special traffic information service. The other cars were routed along the shortest paths using Dijkstra's algorithm. By a route we considered any pair of nodes (source, destination). For each of the 3306 possible routes we compared the average travelling time between the four types of vehicles with the departure in the same time interval. Our first question was if the dynamic routing makes sense in general case and not only when accidents happen. In The Netherlands at rush ours almost all the roads show delays and the possible alternatives to a route might also be congested. Also, we were interested if the current systems that get traffic update every 30 minutes are not already efficient. We wanted to know which is the impact of a higher information update frequency and if it is any space left for our algorithm to improve?

In Figure D.5 we display for the three 'smart' strategies the improvement compared to the static routes. In average for 50% of the routes, faster alternatives were found by the routing systems. We considered that a route was improved when, at a specific point in time, the selected alternative was faster then the initial route. Also, we recorded only the best score difference obtained during the simulation time. The number of deteriorated routes was very low and was depending of the granularity of the time intervals used by the algorithms.

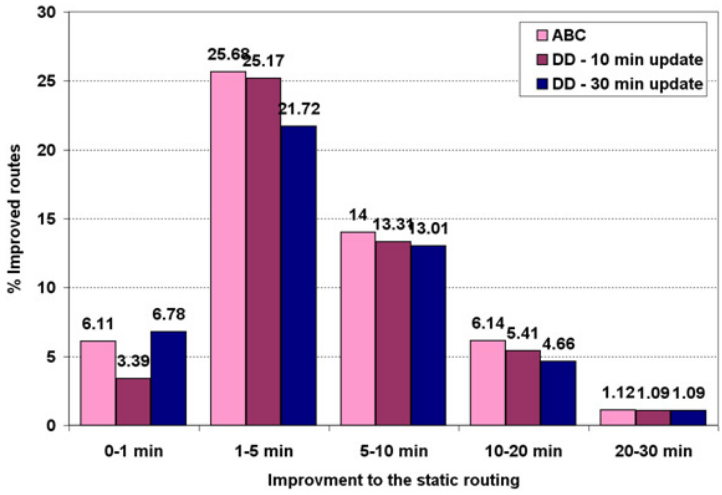


Figure D.5: The difference between dynamic and static routes.

The best score was obtained using ABC algorithm, with 53% of the routes improved. For 6.11% of the cases, an insignificant improvement of less than one minute, is gained compared to the static path. 26.68% of routes suggested by the ABC were up to 5 minutes faster than

the static ones. An improvement of 5 to 10 minutes was noticed in 14% of cases and of 10 to 20 minutes in 6.14% of the situations. For 1.12% of the routes the improvement was up to 30 minutes. If we consider that The Netherlands is a small country and most of the selected routes are shorter than 150 km, then using the dynamic routing system is a real benefit.

Let us focus now on two particular cases. First the A2 motorway, which connects Eindhoven to Amsterdam. It is the most congested corridor in The Netherlands. Here we selected for a detailed discussion the route between the junctions De Hoght, situated at west of Eindhoven, and Muiderberg, positioned east of Amsterdam (Figure D.6). The shortest path has a length of 104 km. There are two main alternatives available. One is to go via Tilburg and follow A27 to Utrecht (130 km). The second, also 130 km long and presented in the figure, is via Nijmegen and Arnhem using A50. For the segment between Utrecht and Amsterdam, the main options are via A2 direction ring Amsterdam, or via A27 and A1.

In Figure D.7 is shown how the travelling time between these two locations has been oscillating during the morning. Between 7:40 and 10:40 the ABC algorithm suggested different faster routes than the static one (Dijkstra). Around 10:00 the ABC suggestion via Nijmegen was in average 15 min faster despite the additional 26 kilometres. The results of the guidance using the current traffic updates were somewhere in between. Increasing the frequency of receiving the traffic information from every 30 minutes to every 10 minutes, didn't show much difference. It can also be observed that for a while, around 9:50 and 11:00, these suggestions were actually slower than the static route. This is because, the alternatives were calculated only using the actual state of the congestion, and the future evolution of the traffic jam was not considered. This was not the case for the routes calculated by the ABC algorithm.

Figure D.6: The shortest and the fastest route from De Hoght to Muiderberg at 9:30

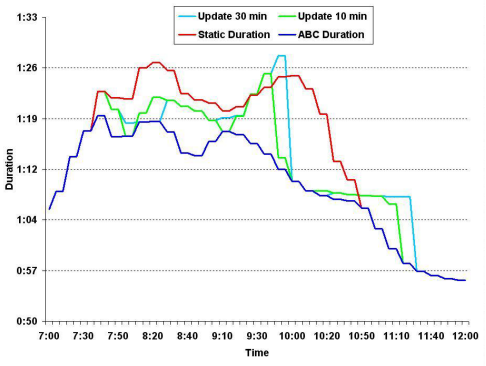


Figure D.7: Traveling time from De Hoght to Muiderberg

For the second particular case we chose as starting point the motorway intersection Empel situated on the ring of 's Hertogenbosch not far away from the Quintiq office (Figure D.8). The destination is Prince Clausplein intersection. It is situated nord from Delft close to The Hague. The shortest way (82 km) is via Utrecht following A2 and A12. But again, many alternatives are available, like going via Waalwijk on A59 then A16 to Rotterdam or via Gorinchem on 15. Also Rotterdam can be passed via east or western ring. The suggestion in the figure is 98 km long.

On this route the ABC suggested better paths from 7:00 to 9:40 (Figure D.9). At 7:50 was recorded the highest difference from the static route: 13 minutes and 39 second. Again we can observe that using the current navigation is not always efficient and sometimes the suggested path proves to be longer in time than the direct route.

5 CONCLUSION

In this paper we discussed the problem of dynamic routing. We adapted the ABC algorithm and we were able to compute the shortest path in time taking into account travel time predictions along road segments. We combined it with historical data from ANWB containing speed

Figure D.8: The shortest and the fastest route from Empel to Prins Clausplain at 8:30

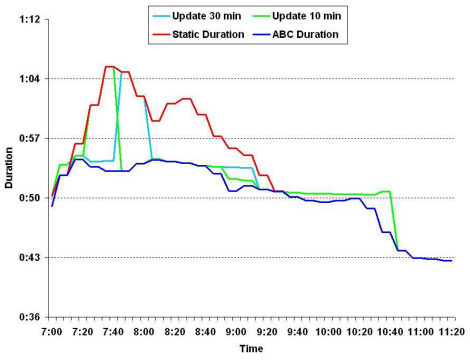


Figure D.9: Traveling time from Empel to Prins Clausplain

measurements along the freeways in the Netherlands. In the rush hours, we see a drop of the maximum speed on most of the roads, so our main question was weather we can reduce the travel time by choosing alternative routes, omitting congestions. It was already proved proved ((Tatomir and Rothkrantz 2004) and (Tatomir and Rothkrantz 2006)) that in case of incidents such as traffic accidents, it makes sense to choose an alternative route. In this paper we demonstrated that dynamic routing results in a reduction of the travel time in general not only in case of accidents. Combining historical information with future prediction, the ABC algorithm is more reliable than just using the current traffic information. In the future we want to analyze how good our travel time prediction is, and to study the influence of different parameters in this feature of the ABC algorithm.

REFERENCES

Ando, Y., O. Masutani, H. Sasaki, H. Iwasaki, Y. Fukazawa, and S. Honiden. 2005. Pheromone model: Application to traffic congestion prediction. *In Engineering Self-Organising Systems*, 182-196.

Chabini, I., and S. Lan. 2002. Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *IEEE Transactions on Intelligent Transportation Systems*, 3: 60-74.

Di Caro, G., and M. Dorigo. 1998. Antnet: distributed stigmergetic control for communication networks, *Journal of Articial Intelligence Research (JAIR)*, 9: 317-365.

Di Caro, G., F. Ducatelle, and L. M. Gambardella. 2005. Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking*, 16(5).

Donati, A., R. Montemanni, N. Casagrande, A. Rizzoli, and L. M. Gambardella. 2008. Time dependent vehicle routine problem with a multi ant colony system. *European Journal of Operational Research 185(3)*: 1174-1191.

Gunes, M., U. Sorges, and I. Bouazizi. 2002. Ara -the ant-colony based routing algorithm for manets. In *Proceedings of the 2002 International Conference on Parallel Processing Workshops*.

Ichoua, S., M. Gendreau, and J. Potvin. 2003. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, (144): 379-396.

Jagadeesh, G., T. Srikanthan, and K. H. Quek. 2002. Heuristic techniques for accelerating hierarchical routing on road networks. *IEEE Transactions on Intelligent Transportation Systems*, 3(4): 301-309.

Kim, S., M. Lewis, and C. White. 2005a. Optimal vehicle routing with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 6(2): 178-188.

Kim, S., M. Lewis, and C. White. 2005b. State space reduction for nonstationary stochastic shortest path problems with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 6(3): 273-284.

Rice, J., and E. van Zwet. 2004. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems*, 5(3): 200-207.

Schoonderwoerd, R., O. Holland, J. Bruten, and L.J.M.Rothkrantz. 1996. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 2: 169-207.

Tatomir, B., and L. Rothkrantz. 2004. Dynamic traffic routing using ant based control. *International Conference on Systems, Man and Cybernetics IEEE SMC*: 3970-3975.

Tatomir, B., and L. Rothkrantz. 2006. Hierarchical routing in traffic using swarm intelligence. *The 9th International IEEE Conference on Intelligent Transportation systems*: 228-235.

Weyns, D., T. Holvoet, and A. Helleboogh. 2007. Anticipatory vehicle routing using delegate multi-agent systems. *The 10th International IEEE Conference on Intelligent Transportation systems*: 87-93.

Wu, C. H., J. Ho, and D. Lee. 2004. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4): 276-281.