TNO report

Communication-based traffic optimisation

TNO Physics and Electronics Laboratory

Oude Waalsdorperweg 63
PO Box 96864
2509 JG The Hague
The Netherlands

Phone +31 70 374 00 00
Fax +31 70 328 09 61

Date
December 2004

Author(s)
R. Nachtegaal

This report has been written within the framework of a traineeship.
TNO-FEL is not responsible for its content, possible conclusions or recommendations.

Contract no	
Sponsor	
Affiliation	
Project officer	
Affiliation	
Classification	
Title	
Managementuittreksel	
Abstract	
Report text	
Appendices	
Classified by	
Classification date	-
Copy no	
No of copies	-
No of pages	85 (incl appendices & RDP, excl distribution list)
No of appendices	-

The TNO Physics and Electronics Laboratory is part of TNO Defence Research which further consists of:

TNO Prins Maurits Laboratory
TNO Human Factors Research Institute

Netherlands Organisation for Applied Scientific Research (TNO)

Contents

Abbreviations

ACP	Application Communication Protocol
CA	Cellular Automaton
DRIP	Dynamic Route Information Panel
EWI	Faculty of Electrical Engineering, Mathematics and Computer Science
GATS	Global Automotive Telematics Standard
GPRS	General Packet Radio Services
GPS	Global Positioning System
GSM	Global System for Mobile communication
GTP	Global Telematics Protocol
IR	Infra-Red
OEM	Original Equipment Manufacturer
RF	Radio-Frequency (Wireless communication technique)
SO	Self-organisation
TNO	Netherlands Organisation for Applied Scientific Research
TNO-FEL	TNO Physics and Electronics Laboratory
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunications Service
WiFi	Wireless Fidelity, the popular term for a high-frequency wireless local area network (WLAN)
XML	eXtensible Mark-up Language

Preface

This report has been written as part of my graduation project at the Faculty of Electrical Engineering, Mathematics and Computer Science at the Delft University of Technology. This project has been carried out at the TNO Physics and Electronics Laboratory in The Hague.

This report will be presented to my graduation committee on December 3, 2004. The committee consists of the following members:

- Prof.dr. A. van Deursen (TU Delft)
- Ir. F. Ververs (TU Delft)
- Drs.dr. L.J.M. Rothkrantz (TU Delft)
- Drs. A. van Lieburg (TNO-FEL)
- Drs. RM Neef (TNO-FEL)

I would like to thank all members for their time and effort, especially F. Ververs, A. van Lieburg and M. Neef. They provided help and advice during the whole project.

Furthermore, I would like to thank all colleagues at TNO-FEL who assisted me with their valuable help, ideas and advice during this project. Special thanks go to Trude Gentenaar, Mark van Lent, Cor van Steijn and Job Tiel Groenestege for volunteering their computers for running my tests in the last weeks. I also like to thank everybody who proofread my report to reduce the number of errors in it.

Rikkert Nachtegaal,
The Hague
November 2004

1 Introduction

This chapter addresses briefly the context of the project. A short introduction of the problem will be mentioned, as well as the scientific and social relevance of it. The chapter concludes with the problem definition in the form of a research question.

1.1 Problem introduction

At this moment, the situation on the road is precarious. The traffic jams are getting longer and longer, millions of Euros are lost because of all the delays. Until now, most of the traffic management is done centrally, with a traffic management centre controlling the Dynamic Route Information Panels (DRIPs) above the roads, the traffic-information that is being sent to radio-stations, and various other means to disseminate traffic information to drivers. When the delays at one particular road or crossing get too high, often a new lane will be created and the situation gets a little better for a few years, but the problems will most likely move to the next or previous bottleneck.

Braess' paradox [9] shows that adding lanes or roads does not always increase the overall road performance. There are situations where adding an extra road leads to more delays and less capacity on the complete network. One of these situations is shown in Figure 1.1, where adding road F3 leads to an increase of traffic on the bridges A and B. The bridges have a lower capacity than the roads, meaning the overall capacity drops when more vehicles choose the new route with two bridges.

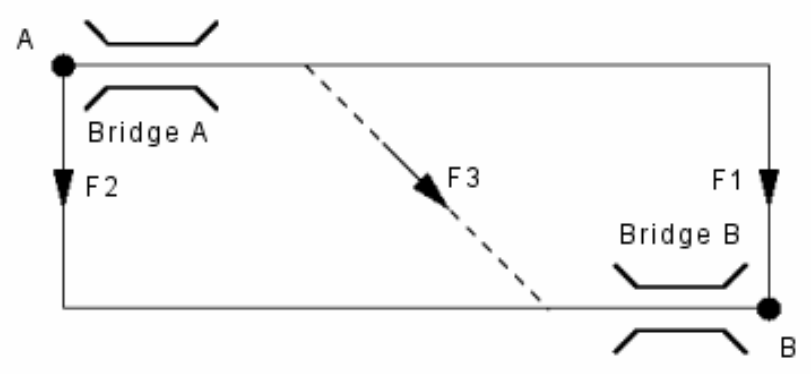

Figure 1.1: Braess' paradox

Another new traffic approach is the “Drive slower, go faster” paradigm [10] advocated by some traffic scientists. This approach demonstrates adding lanes is not as necessary as it seems for increasing the capacity of the roads and the flow in cities. The main objective of this new theory is a new road design: decreasing the cruising speed of traffic to 40 km/h, decreasing the width of the roads to one lane, making a large median strip and removing a lot of the traffic lights. This will lead to a better flow with less waiting time for pedestrians to cross the road, safer traffic and the roads will fit better in the environment.

These two examples indicate that the solution to increasing the road capacity can be found in a completely different approach. Instead of just adding more lanes or more roads, the existing capacity should be used better. One of the approaches is: let traffic manage itself using local interactions. The basis for this study is adding communication between the traffic participants, for example by local interactions. From biology, striking examples are known of well co-ordinated and intelligent group behaviour. Typically these examples demonstrate that even relative non-intelligent organisms like termites can already handle very sophisticated tasks in a well co-ordinated manner. Looking at these examples and other natural mechanisms for co-ordination, the question raises into what extend these mechanisms might be useful in dealing with today’s traffic challenges like congestion.

And although all drivers may have access to communication-technologies in the future, there are still many steps ahead before these technologies will be common. Rather then assuming broad availability of these technologies, we like to consider the case in which just a small group of drivers has access to the information. This can be groups like taxi-drivers, public transport (buses), trucks (cargo transport) or car manufacturers (e.g. BMW, Ford, Renault).

1.2 Research question

The previous introduction can be summarised as follows: The continuous increasing amount of traffic can not be compensated by increasing the number of roads or the number of lanes. The solution to increasing trip times may be found in another direction. Maybe the traffic can be distributed better if drivers have access to more information about the global situation on the roads. This leads to the following main research question:

> *Can we optimise the performance of traffic in general, or certain individuals in particular, by adding (local) communication between a (small) amount of individual drivers or their vehicles?*

The local communication, without global network information centre, sounds like self-organisation principles like those present in the animal realm. Because there are many forms of self-organisation, a brief introduction to this topic is included in this report.

The traffic will be modelled in a software simulator; mounting real devices on vehicles is too expensive and too time-consuming. We will describe the architecture and usage of this simulator.
To test for feasible ideas, the current state of affairs in the traffic market is examined. The ideas of car-manufacturers, universities, research departments and other actors in the traffic domain need to be known, as well as the technical issues involved with employing local vehicle communication.

This leads to multiple issues, which will be answered by an investigation in literature in the preliminary study [11]:

1. *What is self-organisation and how is it being used?*
2. *What is the best traffic-modelling technique, making self-organisation possible?*
3. *What are the ideas of the market about potential applications of vehicle to vehicle communication?*
4. *What communication protocols and techniques are best suitable as means to make self-organising behaviour possible?*

The main research question however can not be answered by researching existing literature, so a road-simulator on which several experiments can be done has to help answering that question. This report describes the requirements, capabilities and the development of this simulator, as well as the set-up and the results of the experiments.

1.3 Experiments

To determine whether the traffic performance can be enhanced by adding communication, several experiments are done by using specially developed software. In the experiments, the density δ and the percentage of vehicles with communication ρ will be changed, to get to know the effects these variables have on the average trip times and the flow (the number of vehicles that reach the endpoint). Also the layout of the network will be changed buring the experiments, to get to know the dependency of the networks on the results.

1.4 Structure of the report

The issues stated in the previous section are answered in the literature survey preceding this report [11]. A summary of that survey and the conclusions of it follow in the next chapter.

After that, a short introduction is given to the software on which the experiments will be done. To be able to create the simulator needed for the experiments, many software design decisions have to be made. These decisions and the design itself will be described after that, followed by an explanation of the implementation of the design. A short user-manual with the capabilities and looks of the road-simulator completes this part.

Finally, the preparations of the experiments as well as the results are given. They lead to the conclusions and recommendations for further research. These conclusions will contain the answer to the main research question.

2 Summary of the research

The four issues stated in the introduction, indicate four different subjects to be researched. In this chapter, a summary of the current state of affairs of these subjects is discussed, which will lead to certain conclusions about the use in of these techniques in future demonstrations or applications.

2.1 Swarm engineering, self-organisation

Swarm Engineering (SE) is the design of systems of simple interacting agents for a specified task. The behaviour of the agents is governed by certain rules. All rules should be local; all results should be global. The swarm engineer will determine the minimal conditions required completing a task and create specific local behaviours designed to satisfy these minimum requirements.

Self-organisation (SO) can be split up in different kinds, based on the underlying theory and application of the SO. In the following paragraphs these types will be explained using real-world examples.

2.1.1 Biological swarming

Biological swarming is like the flocking of birds, or the schooling of fish. This type of self-organisation implicates motion. The individual animals do not know the direction they need to go, but they have a general idea. They have a preference to go with the largest group, when the group goes in roughly the same direction they want to go. So if the group deviates too much from their own idea of the correct direction, they will choose their own route. But other animals in the neighbourhood will adjust their direction according to this individual. This way, the flock will stay together and go in the average direction. This is a very reliable way to find the destination, because the average direction has much smaller deviation than the direction of a single animal.
A second advantage of travelling together is efficiency. A bird can fly with 30% less effort if it is flying behind another bird.

Example of applied biological swarming

Pattern recognition [7]. This is based on autonomous agents who have a preference for ‘sitting’ on an edge. While sitting there, they emit pheromones to tempt other agents. The amount of emitted pheromone is based on the sharpness of the edge. This way, the agents will move to the sharpest edges. This effect is shown in Figure 2.1.

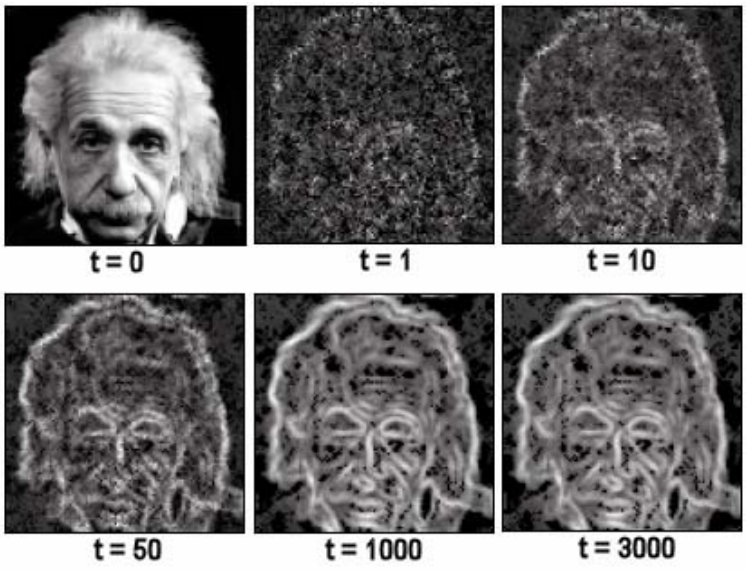

Figure 2.1: Pattern recognition by autonomous agents

2.1.2 Ant-based routing

Opposed to swarming, ant-based routing does not utilise direct interaction (birds can see each other) but indirect interaction (the communication is done via the environment). Ant-based routing is based on trails of pheromone; a chemical fluid emitted by ants when they find food somewhere. Figure 2.2 shows this principle [2], [3], [4], [5].

1. The ants have no idea which route will be the shortest, so both routes have an equal chance to be picked by an ant.
2. The ant on the shortest route will be at the food earlier. That means he will return to the heap quicker. Because both routes have the same amount of pheromone, he will pick the route he just walked.
3. The shortest route now has twice as much pheromone as the longer one. The third ant will choose biased for the shortest route.

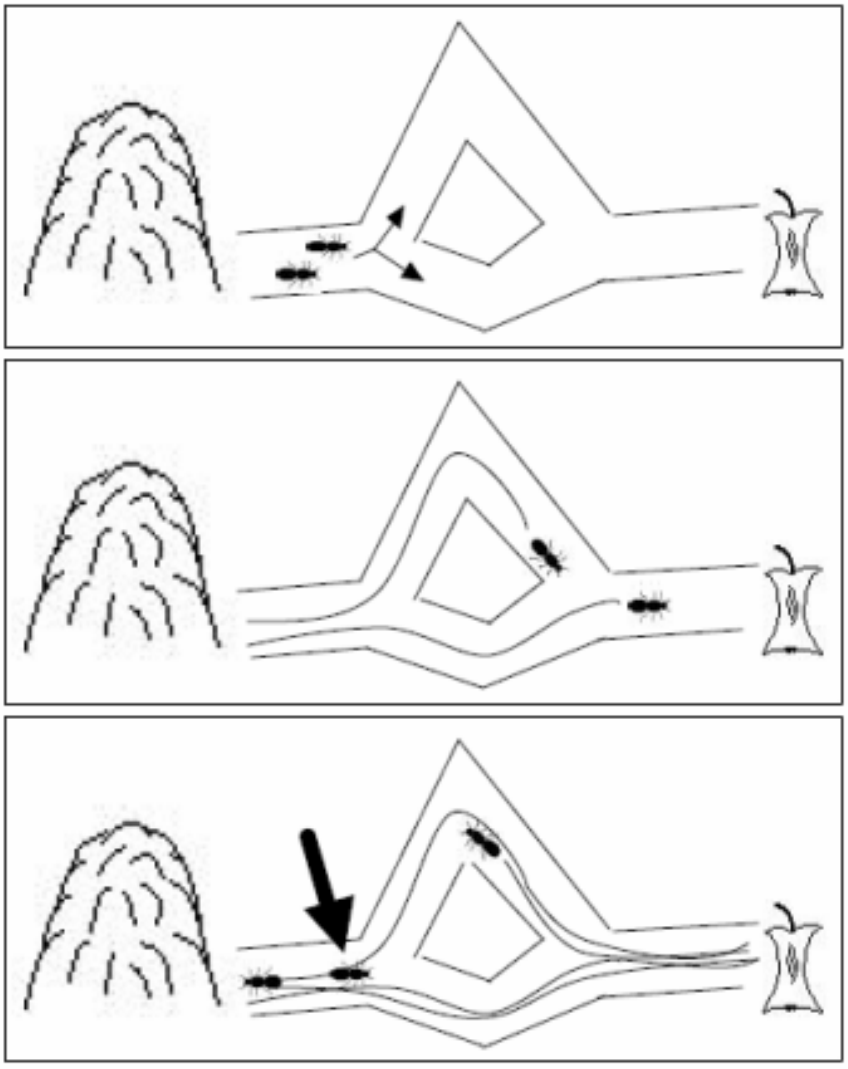

Figure 2.2: Ant-based routing, with the most pheromone on the shortest route

Example of applied ant-based routing

Dynamic telecom routing [3]. By releasing agents in a telecom network that are behaving like ants looking for food, the shortest routes can be found without too much computational power.

2.1.3 Neural networks

A neural network, e.g. the brain, is a highly dynamic network of neurones, adapting to changes in the demand for particular types of neurones and functionality. This example of self-organisation is not about knowledge, but about behaviour. The function of a single group of neurons depends on the demands and current capacity of the system as a whole. So when a single group of neurons detect that there is a higher demand for another function, it can change its behaviour to fulfil this new demand.

Examples of applied adaptive neural networks

- Paint-robots in a car-factory. Switching colours based on the behaviour of ants; switching tasks when needed. Every paint-robot has a preference for a specific colour, but when the amount of waiting cars for another colour is high enough, the robot switches to increase the capacity of that colour. [6]
- Google's pagerank. Google gives all pages a pagerank based, among other things, on the number of links to that page. When a page has more references to it, it will get a higher pagerank thus coming higher in the search results. This way, more people will see the page and if it is helpful they will link to it. It is a self-strengthening process, without centralised control. [31]
- Slashdot. On Slashdot.org, registered visitors have the possibility to react on news items. Other users can moderate these postings, with moderator-points earned by high-rated postings. So, by posting good postings, one earns more points and thus a higher level of authority. This is a very dynamical hierarchy, because when people don't post regularly they will lose their authority-points. When new top-posters arrive, they will be awarded with a higher level of authority automatically. [32]

2.1.4 Conclusion

Swarm Intelligence is a shift in mindset from centralised control to decentralised control and distributed intelligence; from predefined solutions that may break down with the first glitch to emergent, self-organising strategies and tactics. It is about making local interactions that lead to global results. These decentralised principles can be implemented in many different disciplines and when looked at it from this perspective a lot of applications can be said to use self-organisation already. For use in this research, ant-based interaction seems to be the best suitable form of self-organisation.

2.2 Traffic models

To show the potentials of communication between vehicles, two possible approaches can be followed. The first approach is a physical demonstration: create the communication devices, mount them on some vehicles and let them drive in the daily traffic. This however will take lot of time, money and resources, so thankfully there is a second approach: model traffic in a computer-simulation and equip some of the virtual vehicles with communication possibilities. This is an accessible way to demonstrate the possibilities of communication, without having to invest too much. In a demonstration, traffic can be adjusted easily, without the co-operation of all drivers. It is also possible to influence traffic, whereas real-life traffic can only be observed.
In the demonstrations, different elements of this study can be addressed and tested, to find the best solution to the problem, so here we have to look into the different traffic models, to get to know the best model suitable for the demonstration.

To model traffic, several completely different approaches exist. Some of these theories are more suitable for high densities and macroscopic oriented, while others may be oriented towards more microscopic properties. Some of these models are developed to model the traffic as realistically as possible, generating global statistics and prognoses. Other models are meant more for showing or demonstrating specific aspects of traffic, without giving a full factual picture of the traffic.

2.2.1 Fluid-dynamical Theories of vehicular traffic

One of the often-used approaches is the fluid-dynamical model. This model describes traffic as a fluid, with a certain flow into the network and the same flow out of the network [13]. Almost no inter-vehicular relations are taken into account. This hydrodynamic, coarse-grained description of traffic flow adequately describes many of the experimentally observed phenomena. For low densities, however, this continuous approximation fails, and one has to refer to computer simulations and/or different phenomenological approaches. The fluid-dynamical theory can generate spontaneous traffic jams, without a real cause like an accident.

Summarising the recent results of the macroscopic traffic models, there seems to be evidence that on- and off-ramps play an important role for a theoretical explanation of synchronised traffic [17]. Nevertheless some experimental features are still not captured by these approaches. E.g. the empirical results show that for synchronised traffic no correlation between density and flow exist, in contrast to the regular patterns of the oscillating states found in simulations of the macroscopic models.

2.2.2 Kinetic theories of vehicular traffic

In the kinetic theory, traffic is treated as a gas of interacting particles where each particle represents a vehicle [13]. Because of the 'vehicular chaos', which is the analogue of 'molecular chaos' in the kinetic theory of gases, this model is expected to be valid only at very low densities where the correlation between vehicles is negligibly small whereas traffic is better approximated as a continuum fluid at higher densities.

2.2.3 Car-following theories of vehicular traffic

In the car-following theory [14], each individual vehicle has an equation of motion, which is the analogue of the Newton's equation for each individual particle in a system of interacting classical particles. The acceleration may be regarded as the response of the particle to the external forces as well as those arising from its interaction with all the other particles in the system. Each driver can only react by accelerating or braking.

The main drawback of this model is the fact that the parameters of the velocity-function have to be determined by 'calibration', fitting the outcomes of the model with empirical data. The reliability depends on the appropriate choice of these parameters. Also, extension to multi-lane traffic is difficult.

2.2.4 Coupled-map lattice models of vehicular traffic

In the car-following models, both space and time are represented by continuous variables. Also the velocity, acceleration and deceleration of individual vehicles are continuous variables. These continuous variables however, need to be discretised with appropriately chosen grids. In the coupled-map lattice approach [15], time is a discrete variable and the other variables of the individual vehicles at time t+1 are formulated as functions of the variables at time t.
The effects of the interactions among the vehicles enter into the dynamical updating rules only through the distance-headway. Just like the car-following techniques, the variables need to be determined by 'calibration' and multi-lane traffic is difficult.

Several different driving strategies have been compared [16] and one of the outcomes was: If the strategies are dynamically changed to allow drivers to adapt to local traffic conditions, the number of jams and accidents decrease but the flow in high densities is reduced compared to fixed strategies.

2.2.5 Cellular Automata models

Cellular Automata (CA) are an idealisation of physical systems in which both space and time are assumed to be discrete and each of the interacting units can have only a finite number of discrete states. In the CA models of traffic the position, speed, acceleration as well as time are treated as discrete variables. In this approach, a one-dimensional lattice represents each lane with each of the lattice-sites represents a 'cell' which can be either empty or occupied by a 'vehicle'. At each time step the state of the system is updated following a well-defined set of rules, which makes it computational very efficient [13]. This is the main advantage of this approach over the car-following and coupled-map lattice approaches.
This approach considers traffic as a self-organising system, where local interactions between drivers mutually and between drivers and the environment influence the global traffic behaviour.

Nagel-Schreckenberg CA model of vehicular traffic on highways

The Nagel-Schreckenberg (NaSch) Cellular Automaton model is a CA, which is updated in parallel according to four basic rules. Several implementations of this model can be found in [18], [19] and [20]. The four steps of this model are shown in Figure 2.3. After the fourth step, the vehicles are actually moved to their new location.

The four steps of this traffic model are:

a. Acceleration: The speed of the vehicle is increased by one if it is below the maximum speed, else the speed is unchanged.
b. Deceleration (due to other vehicles): If the distance in front is smaller than the current speed, the speed of the vehicle is reduced to the number of free space in front.
c. Randomisation: If the current speed is more than zero, the speed of the vehicle is decreased randomly by one with probability p.
d. Vehicle movement: Each vehicle is moved forward according to its new velocity determined in rules 1 to 3.

The first step describes the tendency of every driver to drive as fast as possible up to a certain speed-limit. The second step is to avoid collisions. The third step is to add randomisation in the model; this step causes the formation of spontaneous traffic jams. The final step is the update-step. Each NaSch model has at least these four rules, but to get a more realistic model more rules have to be added.

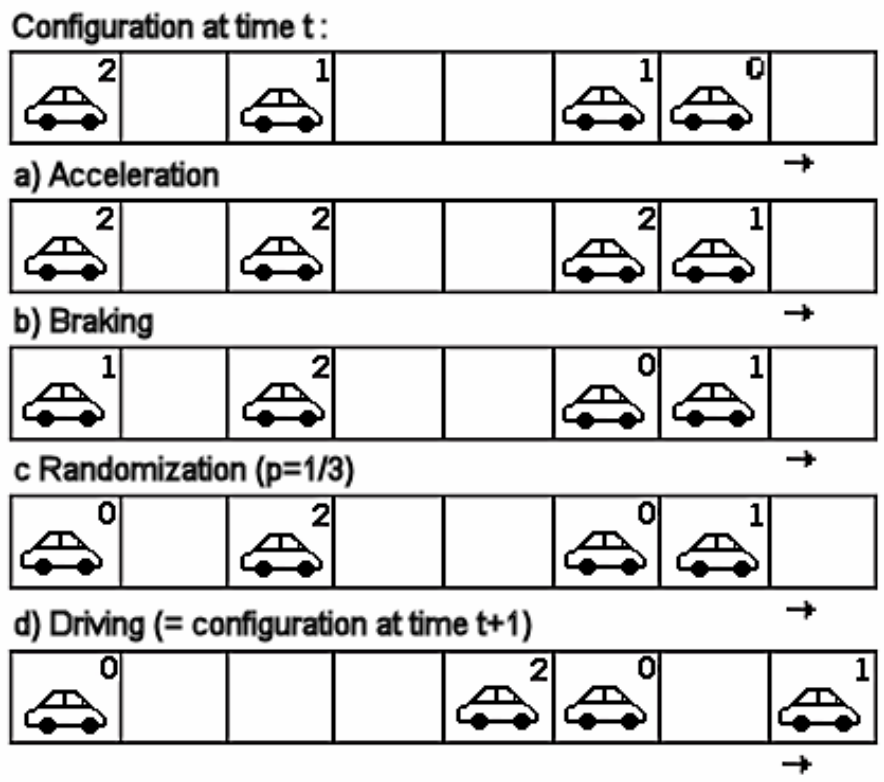

Figure 2.3: The four steps of a Cellular Automaton

Gray and Griffeath four-parameter CA

Lawrence Gray and David Griffeath [22] have done an extensive study into highway traffic, using a cellular automaton. They made a simplified version of the NaSch CA; it is a four-parameter CA-model to identify three different phases of traffic:

- Free flow, where everyone can drive as fast as they want.
- Synchronised flow, most drivers can not go as fast as they want, but they are still moving.
- Stop-and-go, drivers have to stop regularly and then drive a little further and stop again.

This study shows the power and ease to describe real-life traffic situations using cellular automata. The four parameters are the probabilities that a vehicle will advance to the next cell given a certain situation on the road around the vehicle. If an empty cell is represented by a 0 and an occupied one by a 1, Table 2.1 shows the four different transition types, each with their own probability of advancing to the next cell. The current location of the vehicle is *x*, with *(x-1)* as the previous cell and *(x+y)* the next *y* cells.

Table 2.1: Traffic Cellular Automaton update-probabilities

transition type	*(x-1)*	*x*	*(x+1)*	*(x+2)*	probability
accelerating	1	1	0	0	α
braking	0	1	0	1	β
congested	1	1	0	1	γ
driving	0	1	0	0	δ

In their study they show several examples of traffic cellular automata, with different values for these four variables. All three phases can be modelled (shown in Figure 2.4), as well as situations in which spontaneous traffic jams can arise. Due to these characteristics this CA-model is very well suited for use in the traffic simulator.

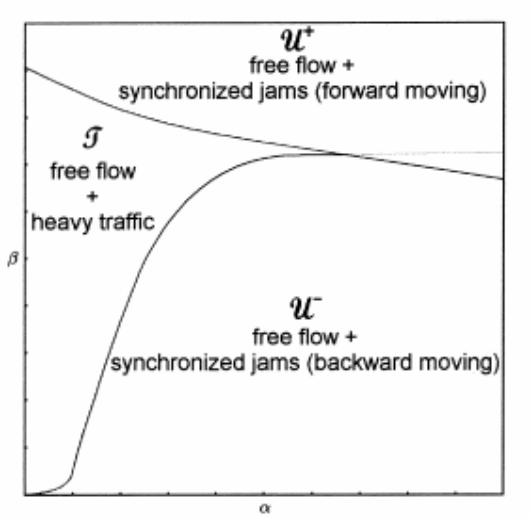

Figure 2.4: Phase portrait from the Gray and Griffeath model

2.2.6 Agent-based modelling

Agent-based models consist of an environment in which entities (agents) interact with each other based on their behaviours (procedural rules) and parameters. In contrast to other modelling principles, where the averaged characteristics of the whole population are considered, in an agent-based model the individual behaviours are considered. Agent-based modelling is also known as entity or individual-based modelling.

There is a great overlap between agent-based modelling and cellular automata. In fact, a cellular automaton is a spatially explicit, grid-based, immobile, individual-based model. Generally, in a cellular automaton all cells are identical, whereas agents can be modelled individually. But given the fact that the cells in a cellular automaton can have different states, the different agents in agent-based models can be considered as different states in a cellular automaton.

The next difference that can be thought of is the sequence of looping through the elements. A cellular automaton proceeds the elements cell-by-cell, where agent-based models proceed agent-by-agent. On parallel-processing hardware this difference is muddied, because the elements are dealt with in parallel without a strict predefined order. So the main difference between cellular automata and agent-based models is the fact whether the simulation is based on a uniform distribution of space (as in a cellular automaton), or based on specific individuals distributed within space (as in agent-based models) [23].

2.2.7 Conclusion

Because of the microscopic orientation of the cellular automata, this seems to be the ideal modelling technique for inter-vehicular communication and the practical use of it. The individual modelling with a number of different vehicles will make it look like an agent-based system, but the cell-based approach will make it a cellular automaton.

The fluid-dynamical and kinetic theories of vehicular traffic are mainly focusing on macroscopic traffic analysis and traffic as a body. Modelling of certain individual vehicles is not possible. In car-following models, individual behaviour can be modelled; all vehicles are being considered as autonomous instances with behaviour depending on external forces on the vehicle. The main drawback of this model is the fact that it can not be extended to a multi-lane situation, making it unsuitable for this study. Coupled-map lattice models can be used, but these models require more computational power than cellular automata because these coupled-map models have continuous variables. So, a cellular automaton seems to be the best choice for individual vehicle modelling and inter-vehicular communication and interaction. In the next stage, there will be demonstrations showing the power and the ease of using cellular automata as model for implementing self-organisation and inter-vehicular communication.

2.3 Intelligent traffic optimisation initiatives using communication

To find out which possible applications of communication between vehicles are worthwhile to implement, several initiatives of car-manufacturers are being looked at. It is very important to know the ideas in the market, because demonstrating favourable concepts, increases the likeliness of acceptation of the ideas.

Initiatives of Ford, BMW and Mitsubishi were investigated, but either none of these car manufacturers is focusing on inter-vehicular communication, or they do not want the rest of the world to know. BMW is at least planning to use the car as a sensor for traffic management, but the other initiatives are only using new sensors in the cars to gather local information which isn't being transmitted and used in a central system, or to other vehicles.

On universities and other research institutes, there are some initiatives in car-to-car communication. For example the graduation work of Ronald Kroon [4], [5] on dynamical vehicle routing using Ant Based Control addresses intelligent vehicle-to-vehicle communication to route vehicles. His work mainly focuses on routing in a city-like network as opposed to the more global highway-network structures this study will focus on.

2.4 Car-to-car communication protocols

To transmit the data between the vehicles, there has to be a communication protocol and a medium capable of transmitting all the data in a relatively short time span. The mostly used communication techniques and bearers are GPS-like devices, GSM signals (GPRS/UMTS), wireless communication (WiFi), infrared communication (IR), Radio-signals (RF).

Researchers at the University of California at Berkeley led by Kris Pister, a professor of electrical engineering, are already working on a smart-dust prototype the size of a small pinhead [7]. It may take many years before this technique will be used, but it is an important technological trend: the spread of sensors and tags. New manufacturing processes, wireless technology and intelligent software are making them ever smaller, smarter and, most important, cheaper. All these sensors will generate a phenomenal quantity of data, which can lead to an information overdose. Letting the sensors communicate with each other and process the data themselves can prevent this. Combining effectors with the sensors can extend such self-organising technology to small intelligent artefacts.

2.4.1 Current situation

At this moment, there are two important protocols used by almost every initiative on car-communication, namely ACP (Application Communication Protocol) and GATS (Global Automotive Telematics Standard). But as the value of telematics lies in the applications (customers do not care about which protocol is implemented in their terminal), the competition between ACP and GATS for market share was also harmful. This competition made the two parties to join their knowledge and create one uniform and universal standard protocol, named Global Telematics Protocol (GTP).

2.4.2 Global Telematics Protocol

The focus within the GTP working group has been to draw the benefits from both protocols focusing on well-proven use cases, while maintaining a simple structure. By developing from an existing standard (ACP) with the enhancements of GATS, GTP is a very reliable standard with years of testing behind it. It supports all communication bearers, not only cellular but also other bearers such as WLAN and DSRC if the market requires it.

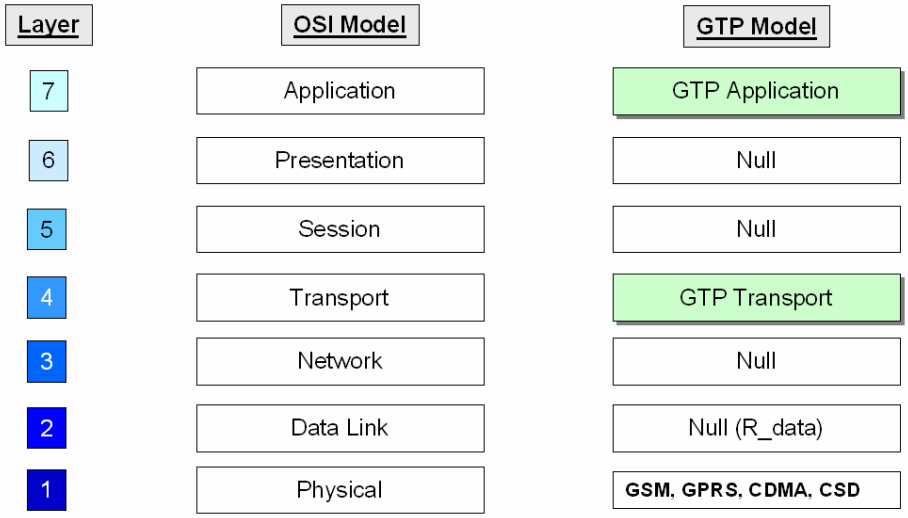

Figure 2.5: GTP compared to the OSI model

GTP is a data transmission protocol that lays over the physical transportation layer and the corresponding data transmission protocols. This protocol can be compared to the OSI model, see Figure 2.5.

The GTP protocol can be applied on almost any transport layer, as long as the transport layer supports the following features:

1. Duplicate packet detection
2. Segmentation and reassemble
3. Packet ordering
4. Error detection with optional correction.

Motorola has offered a transport that is optimised for small bandwidth bearers, such as the SMS (with a bandwidth of 75 baud typically, sometimes even worse), but can also be used for other ones. The GTP group has provided the encoding specifications of the GTP Application Protocol, which describes various kinds of applications with use cases.

The GTP is a coalition of a number of companies in different sectors. From the automotive sector, Adam Opel, Audi, BMW, FIAT, Ford, Renault, Toyota Motor Europe and Volvo are involved. The Service Industry Sector is represented by Cofiroute, Dekra, Gedas, Mizar, NavTech, Orange, PTV, T-Mobile Traffic, TeleAtlas and Vodafone PASSO. Also the Terminal Industry Sector is involved, with the following companies: ACUNIA, Alpine, Ericsson, Mitsubishi, Motorola, Nokia, Panasonic, Philips Semiconductors, Pioneer, Robert Bosch, Siemens and Thales. [27], [34]

2.4.3 Conclusion of the communication protocols

Because of the fact that this protocol (GTP) will be a market-wide protocol with almost every OEM relying on it, GTP can be tested in the demonstrations. GTP is aiming for low-bandwidth, existing physical transport protocols and techniques. This means it is very suitable for using it in vehicle-to-vehicle communication, making local interactions and therefore self-organisation technically possible.

2.5 Conclusion of the research

The main research question is:

> *Can we optimise the performance of traffic in general, or certain individuals in particular, by adding local communication between a small amount of individual drivers or their vehicles?*

Because of the fact that this main question could not be answered in a literature survey, the focus of the literature survey was on answering the issues as stated in chapter 1. These issues were:

1. *What is self-organisation and how is it being used?*
2. *What is the best traffic-modelling technique, making self-organisation possible?*
3. *What are the ideas of the market about potential applications of vehicle to vehicle communication?*
4. *What communication protocols and techniques are best suitable as means to make self-organising behaviour possible?*

In paragraph 2.1, self-organisation was explained, with a short overview of possible applications of considering certain situations in a decentralised way. Several implemented examples of self-organisation techniques were discussed to show the power of this approach.

The fact that traffic is a concerted action between thousands of individual drivers, implicates that self-organisational principles can be used on it. Implementing the traffic simulations on a cellular automaton or an agent-based system links up well with the decentralised approach in traffic modelling.

At this moment, the market is also working on ideas adding communication to traffic participants, except they are mainly focusing on accident prevention and individual data mining. This study is aimed more at using as many cars as possible for the data mining to get the information needed. In the demonstrations the opportunities of this approach will be shown.

At the moment there are two communication protocols in use, but they work together to form one uniform protocol. This protocol aims for low-bandwidth communication, what is needed for the communication between vehicles. So this may be eventually a good protocol to use.

In the next chapters the traffic simulator that will be built for this research will be introduced. By doing several experiments using the simulation environment, the main research question will be answered.

3 Software requirements

In the previous chapters, theory and literature have been discussed, leading to answers to the issues stated in section 1.2. The main research question cannot be answered with the literature itself, but it has to be answered by doing experiments and simulations.

The experiments can not be done in a real-life situation with actual vehicles, but testing them in a traffic simulation environment is feasible. Within TNO, there is a simulation environment named 'Paramics' available. The following definition describes what it is and in what cases it can be used [36].

> *Paramics is a suite of microscopic simulation modules providing an integrated platform for modelling a complete range of real world traffic and transportation problems. Paramics modules work together to improve usability, integration, and productivity allowing users, and their clients, to get added value from the modelling process. Paramics is fully scaleable and designed to handle scenarios as wide-ranging as a single intersection, through to a congested freeway or the modelling of an entire city's traffic system.*

According to this description, the simulation suite seems to be suitable for this project. But it is expected that it will be too time-consuming to use and implement it. So the experiments will be carried out on a simulation environment that will be developed especially for this project.

Before the software can be developed, the requirements on the functionality and the usage of the software have to be known. The requirements can be divided in a set of functional requirements on the user-interaction and a set of technical requirements. The user in this case is the person who will use the software to simulate traffic networks.

3.1 Functional requirements

The software has to satisfy the requirements stated in this paragraph. They are divided into two types. First the general functional requirements for the software are stated.

The simulation software environment must contain at least the following functionalities.

- The software must have command-line options, to adjust some of the variables at the starting-time.
- The software must have random endpoint determination for the vehicles.

- The input variables of the simulation have to exist of at least these properties:
    - The network with the nodes and roads and their characteristics.
    - The duration of the experiments.
    - The proportion of vehicles with the various communication techniques.
    - The density of the roads, meaning the number of vehicles in the demonstrations.
- The output of the simulator has to contain at least the following data, split up for the different ways of communication:
    - The flow of the network (number of vehicles passed).
    - The trip times of the vehicles.

During the development of the software, it became clear that a user-interface would be very handy. It had to be a very clear, user-friendly and easy interface without too many features. This interface has to meet certain requirements. An overview of these requirements is given here.

- It must be possible to turn the graphical output off.
- It has to be possible to add, move, or delete nodes from the network.
- Adding and deleting roads must be possible.
- The weight of each node has to be adjusted.
- The speed of each road has to be individually adjusted.
- The ratio of vehicles with different communication techniques has to be adjustable during the simulation.
- The graphical user-interface must have the possibility to use a grid, to align the nodes and roads nice and easily.
- The user has to have the possibility to choose the network at the start, but it also has to be possible to change the network during runtime.
- The update-speed (the number of updates per second) must be adjustable, preferably during run-time.
- The differences between different groups of vehicles must be made visible in at least the following variables:
    - Average speed
    - Trip times
    - Choices per node
- The vehicles' colouring must be based on destination or on the type of communication, to distinguish them on the roads in the simulator.

Besides these requirements on the overall functionalities of the software and the user-interaction, there are also some requirements on the technical domain which have to be met.

3.2 Technical requirements

The several technical requirements on the software are summed up in the following list.

- The networks have to be stored in and read from an XML format, the definition can be found in section 4.4.
- The output values have to be in CSV format (comma separated values), for easy integration in data-analysis software.
- Every road is a cellular automaton (see section 2.2.5) using the model of Gray and Griffeath [22].
- There has to be individual routing, meaning that each vehicle must have its own network with individual determined weights.
- The network will be stored as a directed weighted graph. A network is internally modelled as a directed weighted graph, consisting of nodes (vertices of the graph) and roads (the edges of a graph).

With these requirements, it is clear what the functionalities and technical needs of the system are. But before software can be built, it has to be designed and specified. In the next chapter, the specifications and design of the simulation environment are described.

4 Software design

With formulating the requirements, it is known ***what*** the software has to do. However, these sets of requirements are not enough to create the software. There has to be a programming design too, in which the structure of the program is worked out more. In this chapter, a more detailed design of the software will be shown. This design is modelled using the Unified Modelling Language (UML), which has been chosen by the Object Management Group (OMG) to be the standard visual modelling language [28]. These UML diagrams will show ***how*** the software has to fulfil the requirements.

4.1 Use cases

The functional requirements as stated in chapter 3 can be visualised by putting them in so-called use cases. Use cases are, as the name implies, based on the user's point-of-view towards the software. The only drawback of these diagrams is the fact that only the graphical form is specified. To get a better insight in the contents and meanings of the use cases, a textual form is desirable. In this report, the textual use cases are created based on the use case templates by Cockburn [30] and can be found in appendix A.

In the first use case diagram, shown in Figure 4.1, the tasks as stated in the requirements for the user-interface (see section 3.1) are shown. The only actor in here is the user of the simulation software. The requirements state seven actions the user must be able to perform, as shown in the following figure.

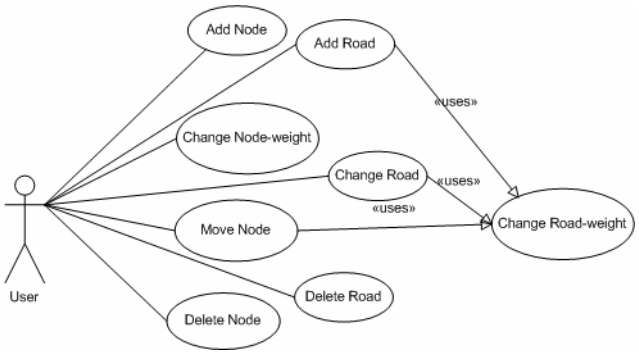

Figure 4.1: Use case diagram of the software-user

Since the software will be simulating a traffic network, it is also possible to make use cases of the traffic participants. The main subject of the research for which the software is being made, was to show and investigate the differences between vehicles with or without any form of communication. This difference is also expressed by the following two use case diagrams. In these diagrams, the actions of the drivers are shown and are split in two to illustrate the different behaviours of the drivers.

The diagram in Figure 4.2 shows the actions of the driver without communication and dynamic routing. This driver decides which route to go at the beginning of the trip, without any reconsideration during the trip.

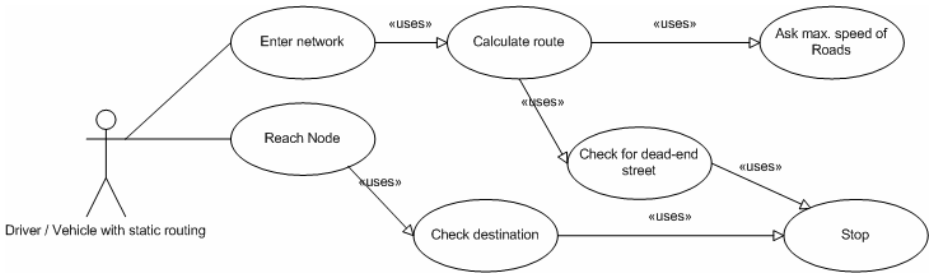

Figure 4.2: Use case diagram of the static vehicles

The use case diagram for traffic-participants with communication possibilities (Figure 4.3) is slightly different. This person can also make decisions about the route after the trip is started. During the trip it will get or ask for information about the state on the roads and use this information to base its decisions on. At every node in the network (a point where one or more roads are connected) there is a possibility to take another route, which might be faster according to the new information.

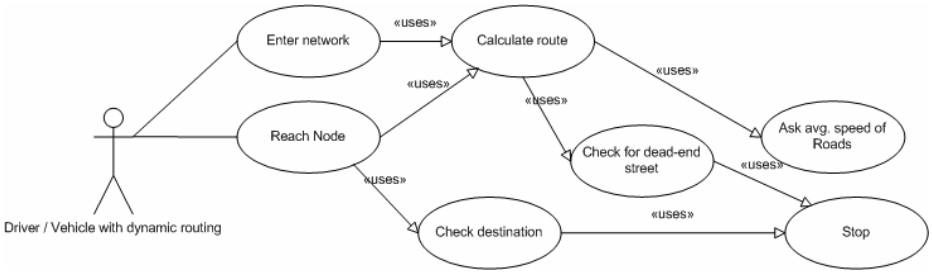

Figure 4.3: Use case diagram of the dynamic vehicles

Now we know what the software has to do, the manner to reach these targets has to be determined. This is what will be explained in the next paragraph.

4.2 Architecture of the software

From the requirements and the use case diagrams, it is known what the software has to do. That knowledge leads to the first global architecture of the simulation software. The software exists of the main program, which controls most of the overall functionality. The other parts of the program can be separated into two parts, on the one hand there is the graphical user interface and on the other hand there are the graph calculating classes. The graphical interface has to consist of at least one window, called a Frame. In most frames there is a drawing board, called a Panel, to draw graphs or the road network on.
As well as the graphics, there has to be an implementation of the network of roads, which will be implemented as a graph. Graphs consist of nodes and vertices, where the nodes represent intersections and the vertices represent roads. The global layout of the software will be as depicted in Figure 4.4.

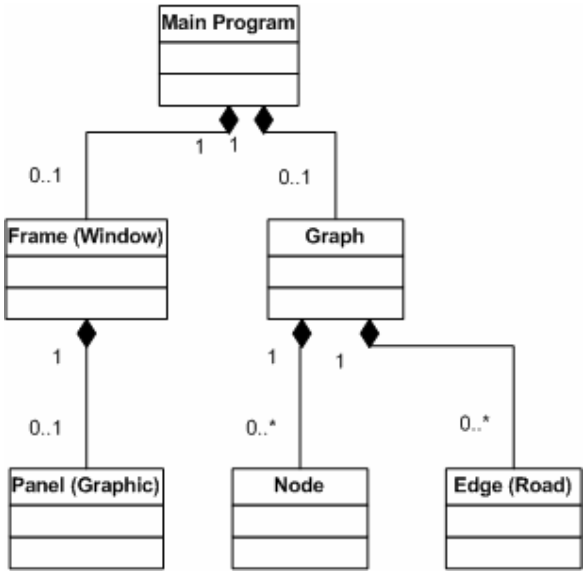

Figure 4.4: Diagram of the global overview of the architecture

This global architecture model will be specified in the next section into several classes and packages, leading to a more detailed model of the simulation software.

4.3 Class model

A class model is used to document the static structure of the software system. It shows what classes exist and how they are related, but not how they interact to achieve certain behaviour. In the previous paragraph the global model was defined, which will be specified item by item here.

Generally, a class-diagram gives a short description of all classes and the static structure between them. The boxes in these diagrams are the classes, with their name in the first field of the box. To keep the figure clear and manageable, the attributes and methods, which can normally be found in the second and third field, are left out. In paragraph 5.2 the classes are worked out in more detail.

At first, the class structure of the frames will be specified. There is one superclass with all the generic window functionalities and variables. The subclasses inherit these functionalities and may add some special features of their own to make the difference. This is shown in Figure 4.5.

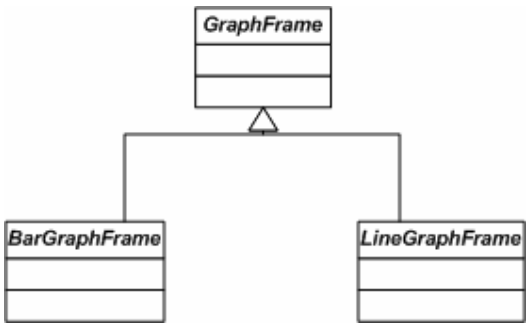

Figure 4.5: Class diagram of the frames

In the panel hierarchy, a similar structure is made. But there will be an extra level in it, because there will be some panels to draw charts on, and some panels to draw roads on. In Figure 4.6 these different types of panels are shown.

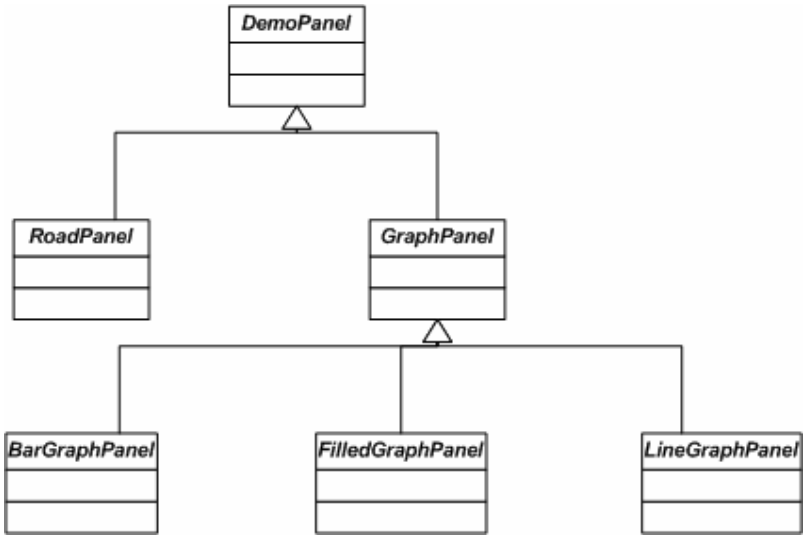

Figure 4.6: Class diagram of the panels

The next item is the road-implementation. Because it will be a cellular automaton, the following structure is quite logical. Each road will be built of several separate cells, which contain one or zero vehicles. This leads to the structure as shown in Figure 4.7.

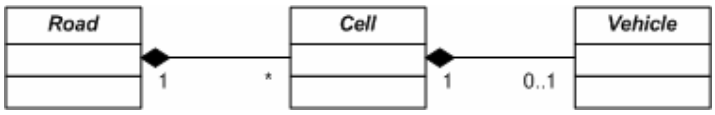

Figure 4.7: Class diagram of the building blocks of the cellular automaton

The last thing that has to be specified is the graph, modelling the network. For this purpose, a special package has been found, called JGraphT [35]. The classes in this package have been used to implement all kinds of graphs ((un-) weighted, (un-) directed, with or without loops) and several techniques to travel the graphs. This extended implementation, as well as the fact that it is free and open-source, made it very feasible to use in the simulation software. Not all classes of the JGraphT implementation have been used, but the structure of the functionality that will be used is shown in Figure 4.8.

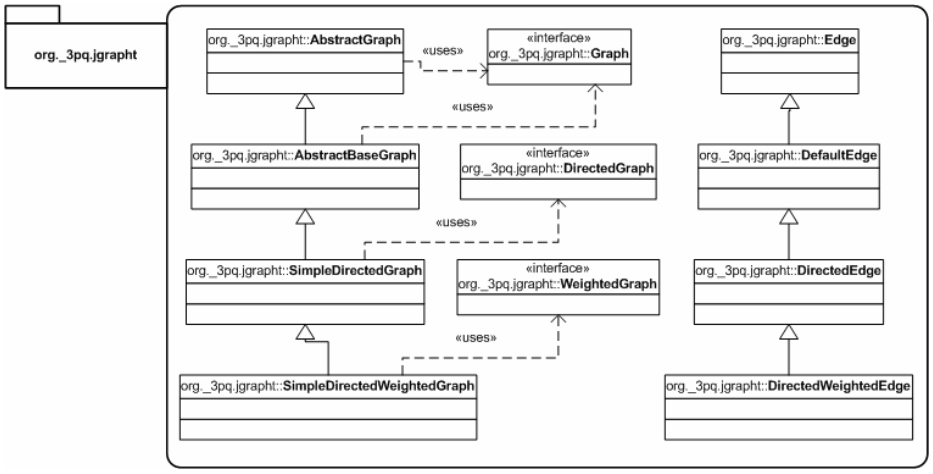

Figure 4.8: Class diagram of the package with the graph implementation

All the specifications described in the previous figures can be put into one big class diagram. Figure 4.9 is showing the complete software structure. In this class diagram the two large round-cornered boxes depict the two packages of the software. The bottom box is the package with the custom-made classes for this project. The topmost box contains the used classes from the package JGraphT [35].

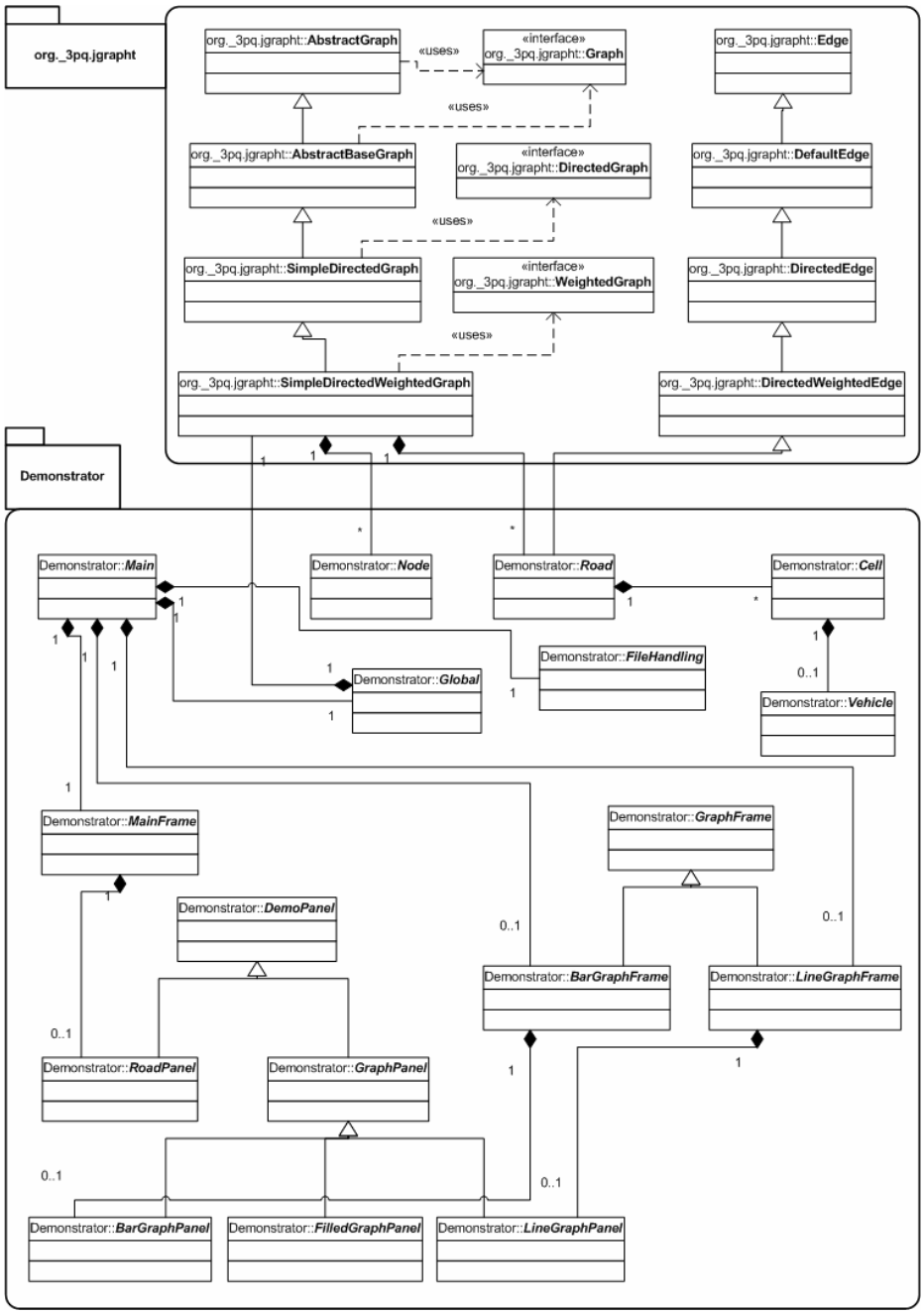

Figure 4.9: Class diagram of the complete system

4.4 Network XML definition

The network and all roads and nodes the network is built of, have to be stored in a file. This file definition has to be easy, extendible and maybe interchangeable between other programs. The XML file format is a text-based extendible format, with easy to understand structures. Every form of information can be stored in these files, as long as they are textual. This makes it a very useful file format for this project.

To store the road networks the following XML definition is used.

```
<network>
	<nodes>
		All nodes
	</nodes>
	<roads>
		All roads
	</roads>
</network>
```

The file exists of one basic block, *<network> ... </network>*, containing all components of the network, being nodes and roads. Those two lines, *All nodes* and *All roads*, deserve special attention. They are replaced by the XML definitions of all nodes and all roads.

The definition thereof is stated below.

```
<node>
	<id>NodeID</id>
	<p>x,y</p>
	<color>RGB color-value</color>
	<newVehicleRatio>ratioOfNewVehicles</newVehicleRatio>
	<weight>weight</weight>
</node>
```

And the roads are defined as follows.

```
<road>
	<beginNode>NodeID</beginNode>
	<endNode>NodeID</endNode>
	<speed>maxspeed</speed>
</road>
```

This way, any network can be stored. There is no limit to the number of nodes or roads, because they are all stored sequentially in their own block (being `<nodes> ... </nodes>` and `<roads> ... </roads>`). An example of such an XML file can be found in appendix B.

5 Software implementation

In this chapter the implementation of the simulation software will be discussed. For example, the technical explanation of the software, as well as the looks and usability of the simulator will be given here. Also some of the bigger problems and solutions to these problems are explained.

5.1 Short software summary

The software is especially developed for this project. It is a road simulator in which the road network can be created or changed interactively. There is no limit on the number of nodes or roads. The network can be saved into an XML-file for later use.

The roads are connected to each other in the nodes. To make sure the roads "know" what happens on the connected roads, the last few cells of the next road are copied to the current road. This way, all roads can check easily whether a vehicle can travel to the next road and with what speed.

The nodes are divided in two types: begin nodes, the ones without any incoming roads and the end nodes. If the network is open (see section 6.1), new vehicles have to be spawned. The chance of a begin node to spawn a new vehicle can be adjusted, from 0% of the time steps to 50%.

The end-nodes have an adjustable weight used to influence the choice of the destination of newly spawn vehicles, varying from zero to ten. Each road has a weight too, which is calculated by dividing the user-configurable maximum speed of the road by the length of it, indicating the time it takes to travel the road.

The vehicles will travel from one of the nodes without incoming roads, to one of the other nodes, randomly distributed based on the weight of each node. The routing of the vehicles through the network is done in two different ways:

- Static: When a vehicle is created, the maximum speed of each road and its length is used to determine the duration of each route. The duration of each road is its weight, used by Dijkstra's shortest path algorithm to calculate the fastest route. These vehicles and the corresponding charts are coloured red.

- Dynamic: Just like the static routing, the shortest route is calculated at the start of the trip, but it will be using the actual average speed on the road instead of using the road's maximum-speed. With the average speed the weight of that road will be determined, which is used by Dijkstra's shortest path method to determine the fastest route. Furthermore, the route is recalculated each time a vehicle reaches a node. These vehicles and the corresponding charts are coloured green.

5.2 Class descriptions

A Class-diagram is a short description of a class. In this section all classes from Figure 4.9 will be discussed with a short summary of the functionality of each class.

5.2.1 Support classes

The support classes are the classes with the global variables, the support functions, the filehandling and the main thread. Also all windows are opened here. These classes are implementing the box "Main" from Figure 4.4.

Main.class

The most important task of this class is the thread. Also the chart windows and the main window are displayed by this. It takes care of the command-line variables and prints the output afterwards.

Global.class

In the Global class, all global variables and functions are kept. This is a static class, because the global information is not instance-dependant. It contains for example variables for the duration of the experiments, the global road network, the time in the world of the simulation. Also functions like Dijkstra's shortest path algorithm, printing debug information and updating the network each time step can be found here.

FileHandling.class

As the name says, this class takes care of the file handling. It stores and retrieves the XML-based configuration files of the networks. The definition of the XML structure can be found in section 4.4.

5.2.2 GUI classes

These classes take care of showing all graphical user interfaces and windows. They are split up in two sets, the windows and the graphical content.

Frames

Each instance of a Frame is a window. They implement the structure shown in Figure 4.5. Because the word graph has two meanings which are both used in this project, it is a little confusing which meaning is meant in the description of the classes. That is why the word ‘chart’ is used to depict a graph in the meaning of a diagram. The classnames however are still named with ‘graph’, leading to somewhat strange effects in the explanations.

GraphFrame.class

This is a non-instantiatable class, only creating a window with the specified title and an instance of a chart panel as defined below. This class has several subclasses which can be instantiated with a specified chart in it.

LineGraphFrame.class

The LineGraphFrame is a subclass of GraphFrame, and is a window containing a line chart (see below).

BarGraphFrame.class

The LineGraphFrame is a subclass of GraphFrame, and is a window containing a bar chart (see below).

MainFrame.class

This is the main window of the simulation application. This window contains the menu to open and close networks, edit the current network and other options. This class takes care of drawing all information, as well as all user interaction like dragging, clicking, moving the mouse and hitting keys on the keyboard. It also takes care of handling all menu actions.

Panels

The following Panel classes are the real charts. The charts are created using off-screen buffering techniques for extra performance. This is the implementation of the structure as stated in Figure 4.6.

DemoPanel.class

This is the main panel class and its only purpose is to provide some generic functions like drawing strings and repainting the background. It also contains the image buffer for the off-screen buffering of the graphics. This makes changing the graphical information much faster; the graphics on the panel are only changed at the end of each update instead of at every minor change.

RoadPanel.class

The roadpanel is the panel used in the main window. It shows the complete network with all vehicles, roads and nodes. It also makes sure that when there is a road from A to B and a road from B to A, they are drawn next to each other instead of on top of each other.

GraphPanel.class
This is the main chart panel class and its only purpose is to provide some generic chart functions like setting the maximum value of the chart and adding values to plot. Charts can be fed with one (single chart) or more (multiple chart) values at a time.

LineGraphPanel.class
This generates a line chart with one or more lines (depending on the input). When only one input variable is given, it will be black, otherwise colours will be used.

FilledGraphPanel.class
This generates a line chart with one or more areas (depending on the input). An area chart is just like a line chart, but the area between the line and the horizontal axis will be filled with the colour of the line. When only one input variable is given, it will be black.

BarGraphPanel.class
A bar chart is a special type of chart. The other charts have a history of the values, but a bar chart only shows the data provided at that very moment. This chart can contain as many horizontal bars as needed. The number of values is a two-dimensional array, so an $n \times m$ chart (with n sources with m values each) is possible. The size of the bars will be adapted immediately.

5.2.3 Core classes
The actual objects of the CA simulator are in the core classes. These objects are more or less like the real objects in the world and implement for example the cellular automaton as shown in Figure 4.7.

Node.class
A node is a connection point of one or more roads. It is stored as a vertex of the graph representing the network. It maintains the history of the number of vehicles passed and the direction of the vehicles.

Road.class
A road is a weighted, directed connection between two nodes, represented by an edge in the network graph. The road is a cellular automaton, so it exists of an array of cells containing the vehicles. The cellular automaton is implemented according to “The Ergodic Theory of Traffic Jams”, by Lawrence Gray and David Griffeath [22], see also section 2.2.5.

Cell.class
A cell is the building block of a road. In each cell there is room for one vehicle at most. This class only keeps track of the fact whether a vehicle exists or not.

Vehicle.class

A vehicle is an object which moves from cell to cell on the road, without notice of the individual cells. If the vehicle uses dynamical routing, it will recalculate the route at the end of a road to see which of the possible next roads seems to be the best choice. Each vehicle keeps track of the time and distance on the current road and in the network, so it can calculate its average speed to transmit it to other verhicles. Each vehicle has either a dynamic or a static route-planning. In a static vehicle, the weights of the roads are determined by the maximum speed and in a dynamic vehicle the weight is defined by the average speed of the vehicles on the roads.

5.3 Software optimisations

During the implementation, some performance issues were encountered. These issues are solved by some software optimisations.

The first and most important issue was a memory problem. Every vehicle has its own road network, because every vehicle had to have individual routing. This causes every vehicle to be quite a large object in memory. In the first tests, every spawn vehicle was a new object and every vehicle reaching its destination had to be thrown away by the garbage collector. The software used enormous amounts of memory because of this, causing the processing-power-consuming garbage collector to run very often. The memory consumption is shown in Figure 5.1, in which the vertical axis shows amount of memory used and the horizontal axis depicts the time.

Figure 5.1: Memory consumption before optimisation (time flows left-to-right)

Figure 5.2: Memory consumption after optimisation (time flows left-to-right)

This issue was solved by implementing a virtual vehicle queue. All deleted vehicles are stored in this dynamic array and all new vehicles are fetched out of it. The results of this optimisation are shown in Figure 5.2. It is clear to see that the amount of memory used increases much slower, causing the garbage collector to run less often.

The second optimisation is in the order of processing the cells of a road. By default, a CA is updated from begin to end. But by processing it from end to begin, the empty spaces in front of each vehicle will be counted automatically. This way, its speed can be adjusted without having to look forward at each vehicle.

The last optimisation is about the choices of a vehicle which road to take next. This choice will be cached if there is no empty spot on the next road. This choice is cached for one time step only, because it caused a grid lock while waiting to get on the next road. Besides that, it is more realistic to reconsider the decision. Real drivers will also reconsider their choice for a specific road when they have to wait too long to get on that road.

5.4 Layout of the user-interface

The software consists of one main screen and several smaller screens next to it, shown in Figure 5.3.
The main screen shows the network with the roads (black lines) and the nodes (coloured pluses). Also the menu for file operations and editing the network is in this screen. At the bottom of the screen, the sliders for the weights, speeds and ratios as explained in section 3.1 are situated.

Next to the screen with the network are three smaller white windows. These windows contain charts about the network or the simulation currently running. In the first screen the average speed of the vehicles can be found, in the second screen the average trip time of the last 200 vehicles is shown. The bottom screen is somewhat different. This screen only contains information when a node is selected in the network. The chart shows the choices vehicles made in the selected node about their next node. It will show the difference between the routing of the vehicle-types very clearly.

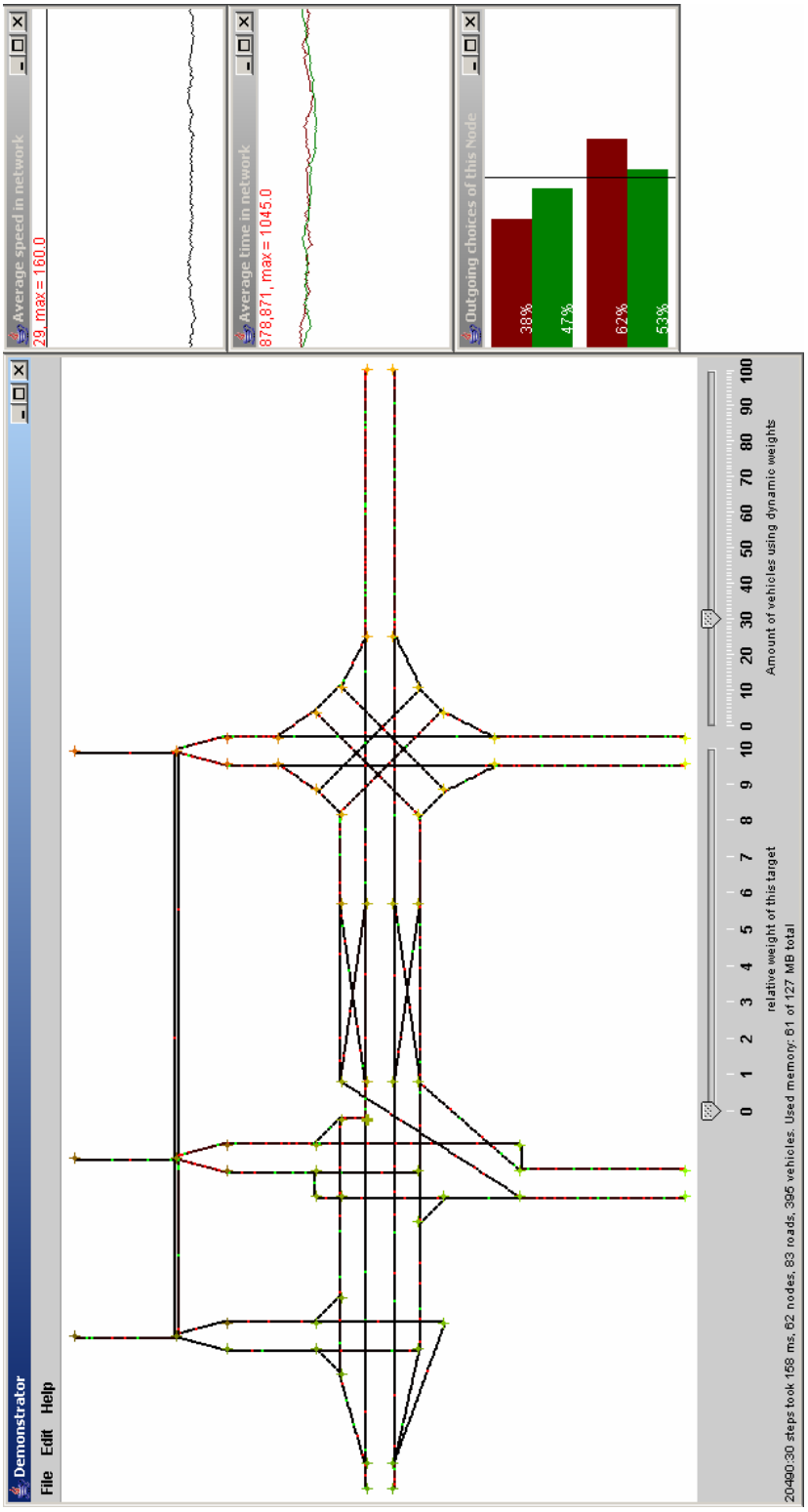

Figure 5.3: Large screenshot of the simulation software

6 Introduction to the experiments

Before the actual experiments can be performed, several issues have to be explained. First the networks to test with are shown. Also some input and output variables are defined here. Further, the duration of each test has to be determined. After these introductions the experiments can be done.

6.1 Network types

The networks can be open or closed to form a ring. In a closed network, the end node is internally connected to the begin node. This leads to a constant amount of vehicles in the network. The closed form is mostly suitable for modelling highway traffic and traffic jams in particular.

An open network is more suitable for city-like networks, with several begin and end nodes. In an open network each node has its own weight of being an end node and each begin node can have its own percentage of new vehicles spawned each update.

All experiments in this study will be done using closed networks. Only in the recommendation in chapter 9 open networks will be compared to the results of the closed networks.

6.2 Networks

More than one network is needed to test with, to determine the influence of the network layout on the results. That is why some different networks have been designed, all with only one begin and one end node. This is done to get dependable outcomes which can be compared to each other. The begin node is the leftmost node and the end node is the rightmost one.

A. **Simple three-road network**: This will be a very simple network, with one begin-, one endpoint and three possible routes to get there (see Figure 6.1). The shortest route is the middle one and that will lead through a virtual city. The speed will be very low. The second route (the upper one) is using the highway which is a longer but faster way. The third possibility is a highway of which the speed has been decreased for the environment (the bottom one). This network consists of 12 nodes and 13 roads with a total of 1174 cells. Each cell is a position on which a vehicle can be, so at a density of 100% there are 1174 vehicles in the network.

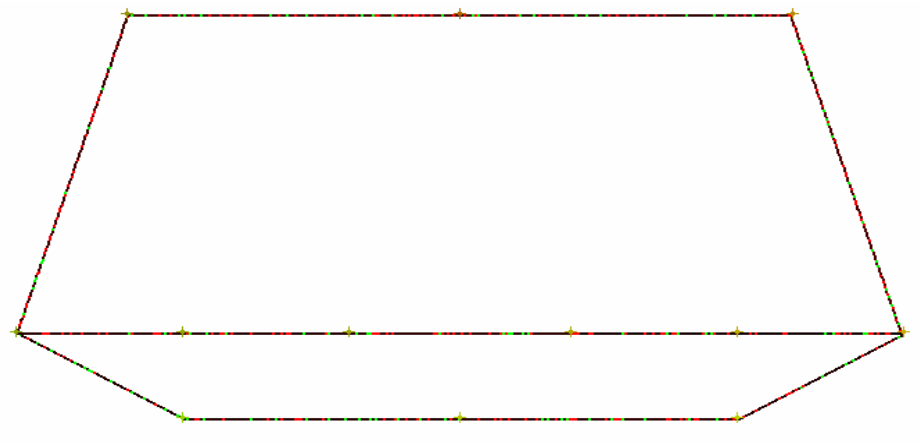

Figure 6.1: Simple three-road network

B. **Extended three-road network**: It is the same network as the simple one, but with several interconnects between the three roads, enabling the possibility to change your route after you chose one at the begin-node. This network is shown in Figure 6.2. The traffic on all roads flows left-to-right; therefore there are no loops in this network. This network consists of 12 nodes and 19 roads with a total of 1696 cells.

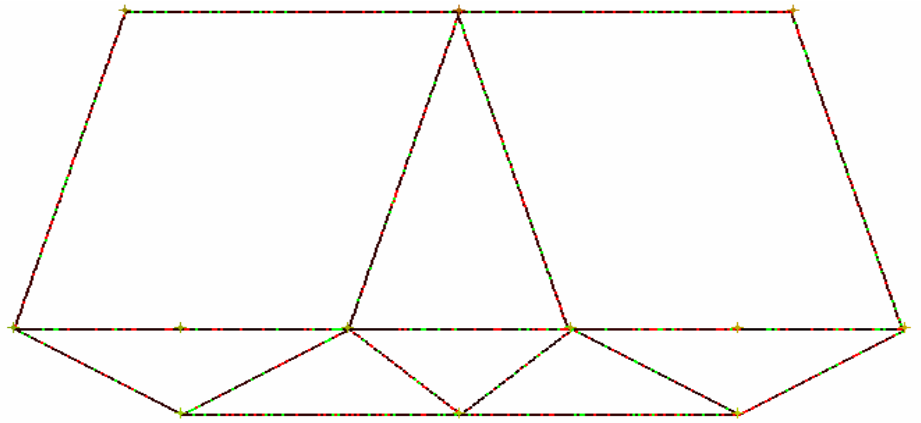

Figure 6.2: Extended three-road network

C. **Extended five-road network**: It is based on the extended three-road network, but with a double slow road in the middle and an extra highway on the bottom. Figure 6.3 shows this network. The traffic on all roads flows left-to-right; therefore there are no loops in this network. This network consists of 18 nodes and 41 roads with a total of 2996 cells.

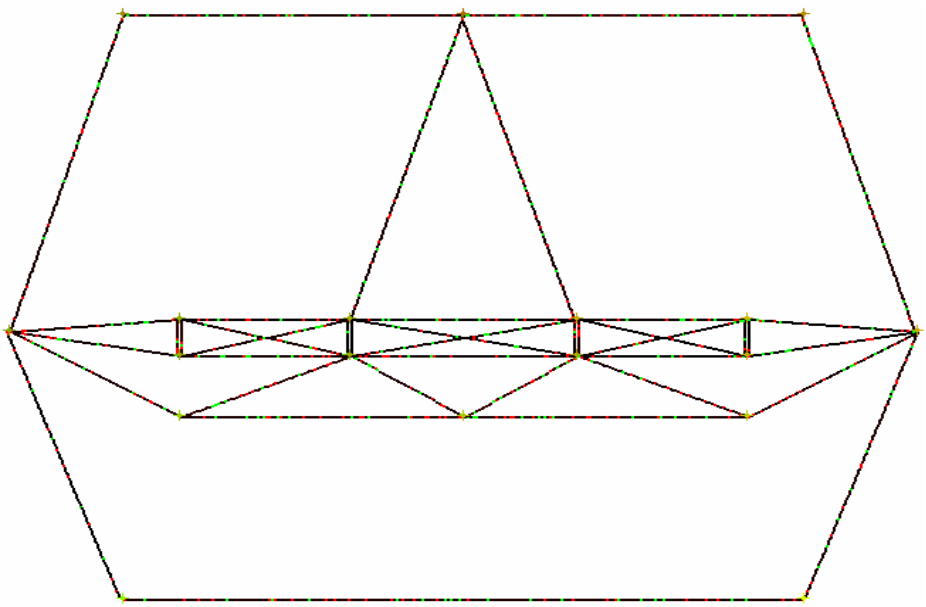

Figure 6.3: Extended five-road network

With these three networks the experiments will be performed. The outcomes of the tests will be compared to find the dependency of the network layout on the results.

6.3 Definitions

For a correct understanding of all figures, tables, outcomes and charts in the following paragraphs, some terms have to be explained.

Cell size

Each cell of a road represents 10 meters in the real world. The vehicles will move a maximum of one cell per time step, which is defined being 160 km/h or 44.4 m/s. That means that a vehicle has to move 4.4 cells per second. The CA model (see section 2.2.5) has a probability of 0.6 for a vehicle to move forward when there is enough space. Concluding, there has to be 4.4 / 0.6 = 7.4 updates per second, so each step in the model means 0.135 seconds in real time.

Confidence Interval (CI)

If independent samples are taken repeatedly from the same population, and a confidence interval calculated for each sample, then a certain percentage (confidence level) of the intervals will include the unknown population parameter. So a “90% CI” means the estimation is at least 90% of the samples for the particular parameter will be in this interval.

The width of the confidence interval gives an idea about how uncertain the unknown parameter is. A very wide interval may indicate that more data should be collected before anything very definite can be said about the parameter.

Density (δ)
The density of a road denotes the "fullness" of it, varying from 0.0 (no vehicles on the road) to 1.0 (every cell of the road is occupied). The roads are being initialized with a certain density, after which the amount of vehicles will stay the same.

Dynamic ratio (ρ)
This is the ratio of vehicles with dynamic routing as fraction of the total amount of vehicles in the network. At 0.0 there are only vehicles with static routing and a $\rho = 1.0$ means there are only dynamic routing vehicles.

Flow
The amount of vehicles travelled from their begin node to their end node within a certain amount of time.

Trip time
The time it takes for a vehicle to get from its begin to its end node. In all networks there is only one begin and one end node. This is done to compare the trip times, when there are more possible begin or end nodes, the trip time of two vehicles can not be compared.

6.4 Initialisation- and run-times

The duration of all tests can be divided in two parts: the initialisation time and the run time. During the initialisation time, the output variables are not measured. Only the outcomes during the run time are taken into account. From a performance point of view, the initialisation time has to be as short as possible to minimise the waiting time. But the actual measurements should not be started before the network is in balance, so the initialisation time needs a certain minimal duration. In the next section the duration of this initialisation will be determined.

6.4.1 Theory about dynamic initialisation times

The network will be started with a certain amount of vehicles (depending on the density δ), at random positions on the roads. The first time a vehicle reaches its end node; its duration will not be taken into account. In that case it has not travelled a complete route; therefore it is unknown how long it would take to travel the complete route. When all vehicles have finished the first incomplete trip, the initialisation period ends. From that moment the trip time of all vehicles reaching their end node will be counted.

The duration of the first incomplete trip depends mainly on the network-layout and the density, so the initialisation time of the experiments can not be one fixed value for all tests. Taking this into consideration, the initialisation times will be dynamically determined per experiment.

6.4.2 Test

To test the theory posed in the previous paragraph, the trip times of all vehicles will be measured and shown in a chart. Every line represents a vehicle reaching the destination; the height of a line is the time it took to drive the route.

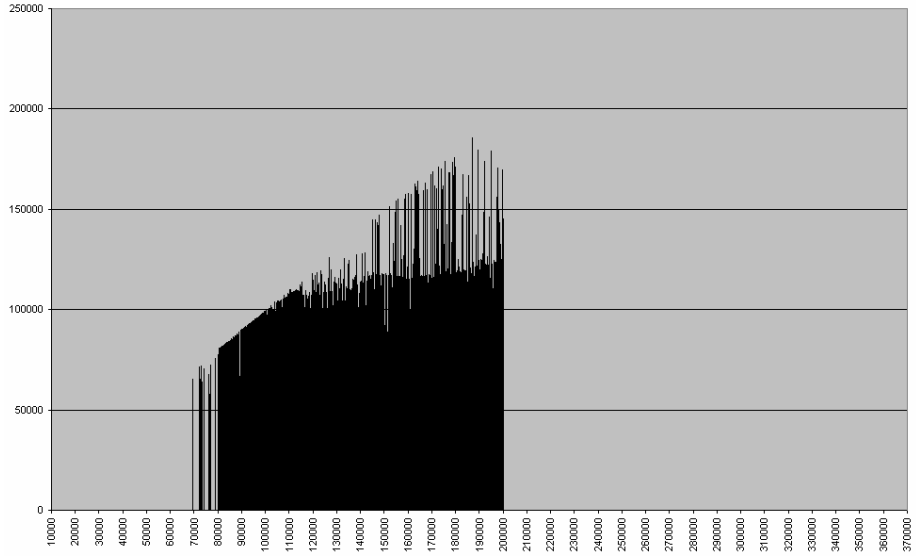

Figure 6.4: Trip times of all vehicles without an initialisation time, no stable trip times in this example

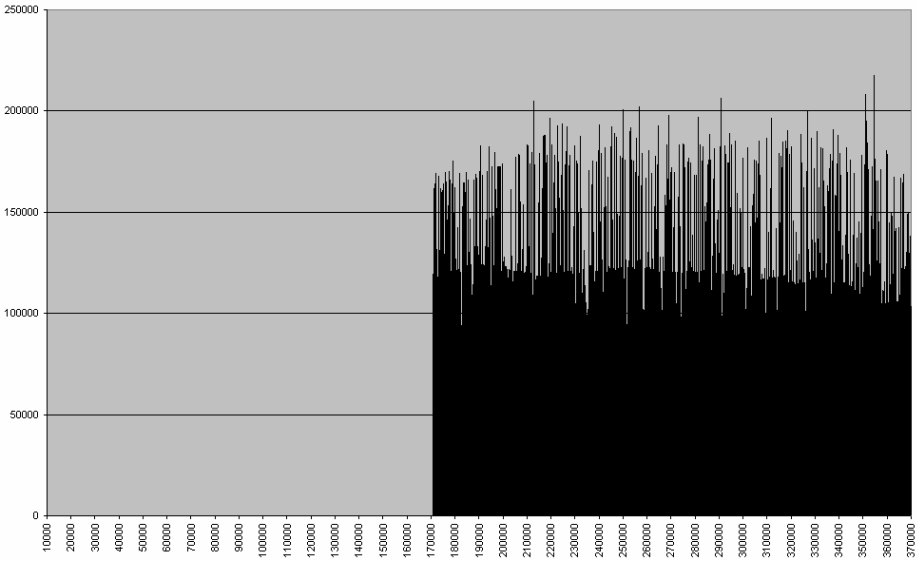

Figure 6.5: Trip times of all vehicles with an initialisation time, leading to more stabilised outcomes

In the first situation (Figure 6.4), there is no initialisation time. Every vehicle which travelled a complete route will be taken into account, so the theory from 6.4.1 is not used. The extended three-road network is used for this example, with a density (δ) of 0.9 and a dynamic ratio (ρ) of 0.0. The test is run for 200,000 steps.

The second chart (Figure 6.5) shows the same situation, but with an initialisation time: the vehicles will not be counted until all vehicles have reached the end point one time. This test is run also for 200,000 steps.

The lines in the first chart do not start before t=70,000 because there has not been a vehicle that completed his route within this time. Note the height of the first line, it is around the 70,000 too.

6.4.3 Conclusion

It is clear that the trip times in the second chart are stable as opposed to the first chart. The theory of 6.4.1 works out quite well, the stabilisation begins about the moment the last vehicle begins its first complete trip (that is the first line in the second chart). So in the experiments this method will be used to determine the initialisation of the network. It will be limited to 100,000 time steps, to prevent an immense initialisation time. The duration of the run will be a fixed amount of time; the vehicles will be counted during 20,000 steps. The complete test will therefore run for a maximum of 120,000 time steps.

7 Experiments

To determine whether dynamic route information has any effect, several experiments are done by using the software described in the previous chapters. The outcomes as specified in section 3.1 will be determined for all networks as described in Chapter 6. The networks will be closed, as explained in section 6.1.

7.1 Results

In the experiments, the density δ and the percentage of vehicles with communication ρ will be changed, to get to know the effects these variables have on the average trip times and the flow (the number of vehicles that reach the endpoint). The trip times and flow will be measured for three groups:

- the vehicles with communication
- the vehicles without communication
- all vehicles in general

To get more dependable values, each test is done 100 times and averaged over these 100 runs. The required computational power for these tests is quite huge; one test of 100 runs can take up to 16 hours of computation on a normal personal computer (1800 MHz).

7.1.1 Trip times

The first variable measured is the most important one for traffic participants: the trip time. It is the average time vehicles need to get from begin to the end point. In the following charts, the trip times are plotted for nine densities. Each line connects all tests with the same ratio of dynamic routing vehicles (ρ). The ratios are tested from 0.0 (no dynamic routing vehicles) to 1.0 (all vehicles use dynamic routing). All data can be found in Appendix C.

Gray and Griffeath [22] did not measure the trip times of their CA, but the speed of the vehicles. The course of the speed plotted against the density is shown in Figure 7.1. To get the trip times, the values have to be inverted, leading to a slightly increasing line. The exact values of this chart will not match the numbers from the experiments, because they tested a single CA and not networks like these. But the graphs from the experiments will resemble this figure.

Figure 7.1: Speed of vehicles in the Gray and Griffeath CA plotted against the density on the road

The three charts depict the data of the three networks from Chapter 6. Each line of the chart depicts a specific value for the ratio of dynamic routing, ranging from 0.0 to 1.0. The density in the network is on the horizontal axis and the trip times are on the vertical axis.

The most striking effect shown in the charts is that a higher density leads to higher trip times. This effect is self-evident, if there are more vehicles on the road traffic jams will arise. The traffic participants can not drive as fast as they want, which leads to higher trip times.

When more vehicles are equipped with dynamic routing, the overall trip time decreases, until the ratio of dynamic vehicles exceeds a certain network-depending maximum. Beyond this value, the trip times will slightly increase again. This effect can be explained by the fact that there are less static routing vehicles, taking the longer static routes. More vehicles will choose for the alternative routes, making them less interesting.

The differences between the tests are small in the tests with the simple network (Figure 7.2), because it lacks decent alternative routes. Once a vehicle has chosen a road, it can not change that decision. That is why the optimal ratio ρ is much lower in that case. In the most extended network (Figure 7.3) the higher amount of vehicles using dynamic routing does not lead to an increase in trip times, as in the three-road extended network (Figure 7.4). The explanation of this difference is that in the five-road extended network the dynamic routing vehicles are always able to find an alternative route, even at high densities.

Another striking effect is the much higher overall trip times in the extended networks with regard to the simple network at the same ratio of dynamic routing vehicles. The initial expectation would be that more alternative routes would lead to a decrease of the overall trip times. This effect can be explained by the fact that all roads in the network are initialised with a certain density. The more roads there are the more vehicles will be created in the network. Because of the fact that the added roads are extra connections between the main routes (compare the networks in section 6.2), the density on those roads will decrease during the experiment. The other roads will therefore contain more vehicles at a certain global density than they would have in the simple network with the same density. The trip times are therefore not completely comparable between the networks.

Because of the fact that the initialisation time is not unlimited, as stated in paragraph 6.4.3, the trip times at high densities can not rise above 120,000. This is why the the lines at ρ=0.0 in Figure 7.3 and Figure 7.4 have a bend and flatten towards 100,000.

Simple three-road network

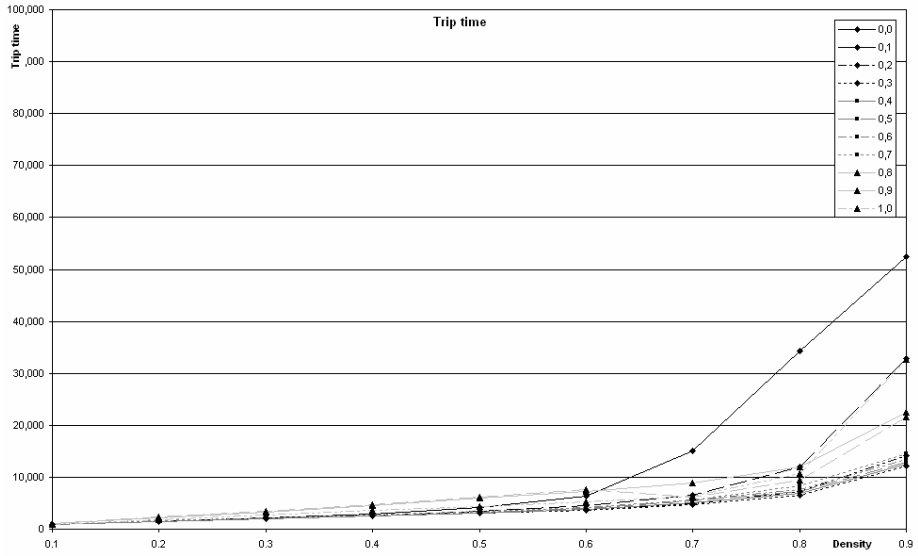

Figure 7.2: Trip times of all vehicles in the simple three-road network

Extended three-road network

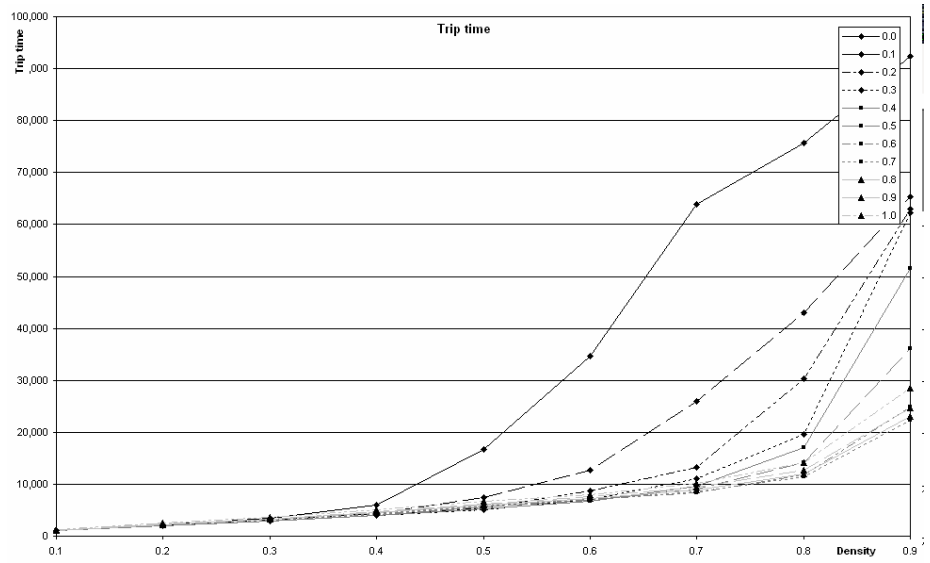

Figure 7.3: Trip times of vehicles in the extended three-road network, with larger differences between the various ratios of dynamic routing vehicles

Extended five-road network

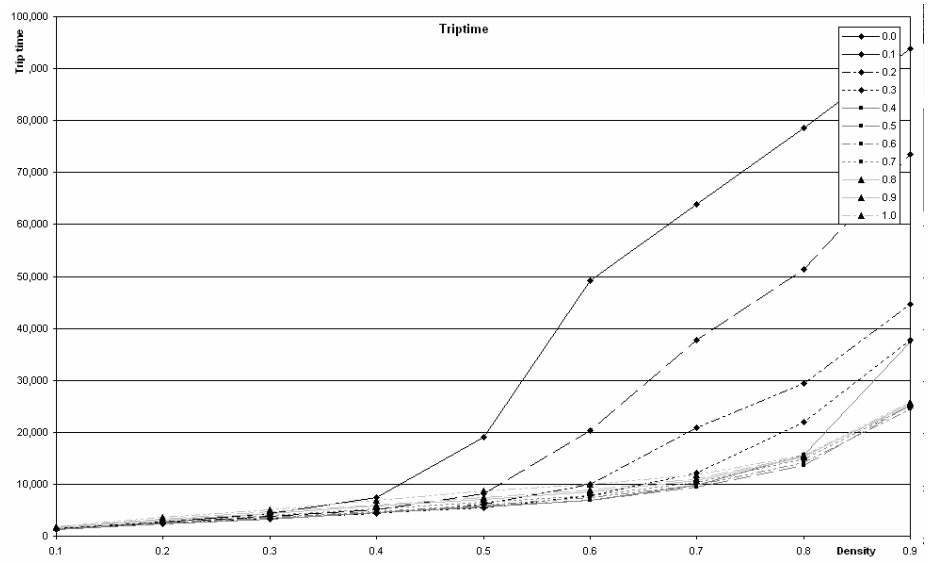

Figure 7.4: Trip times of vehicles in the extended five-road network, with even more diverging lines because of the increased routing options

7.1.2 Flow

The second variable is important for the traffic scientists, because it tells something about the performance of the traffic in general. The higher the flow, the more vehicles are able to reach their destination in a certain amount of time. In the three charts in Figure 7.6, Figure 7.7 and Figure 7.8 the flow of the three networks are shown. Each line of the chart depicts a specific value for the ratio of dynamic routing, ranging from 0.0 to 1.0. The density in the network is on the horizontal axis and the flow is on the vertical axis.

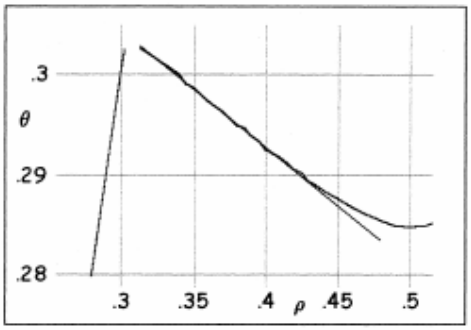

Figure 7.5: Flow in the standard CA of Gray and Griffeath

The flow of the CA made by Gray and Griffeath [22] is shown in Figure 7.5. The exact values in this chart will not match the numbers from the experiments, because they tested a single CA and not networks like these. But the graphs from the experiments will resemble this figure.

In the three figures (Figure 7.6, Figure 7.7 and Figure 7.8) the effect of the dynamic routing vehicles is very clear. One of the most remarkable effects is the course of the lines. At higher ratios the throughput begins a little lower, but the sharp decrease in every line moves slowly to the right when adding more vehicles with dynamic routing. This means when more vehicles use dynamic routing, there are more vehicles that can keep driving at higher densities. But not only is the sharp decrease moving to the right, it is getting less sharp. The amount of decrease on higher densities is decreasing; meaning the flow of the network is steadier.

The differences in flow are the largest in the most extended network, where the vehicles have the most alternatives.

Simple three-road network

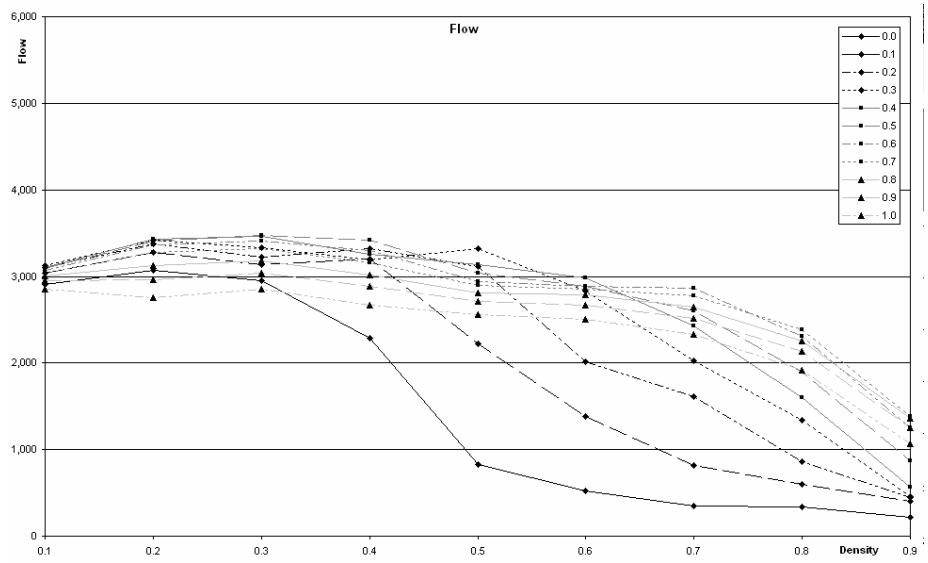

Figure 7.6: Flow of all vehicles in the simple three-road network. Notice the stabilising lines when more vehicles are equipped with communication.

Extended three-road network

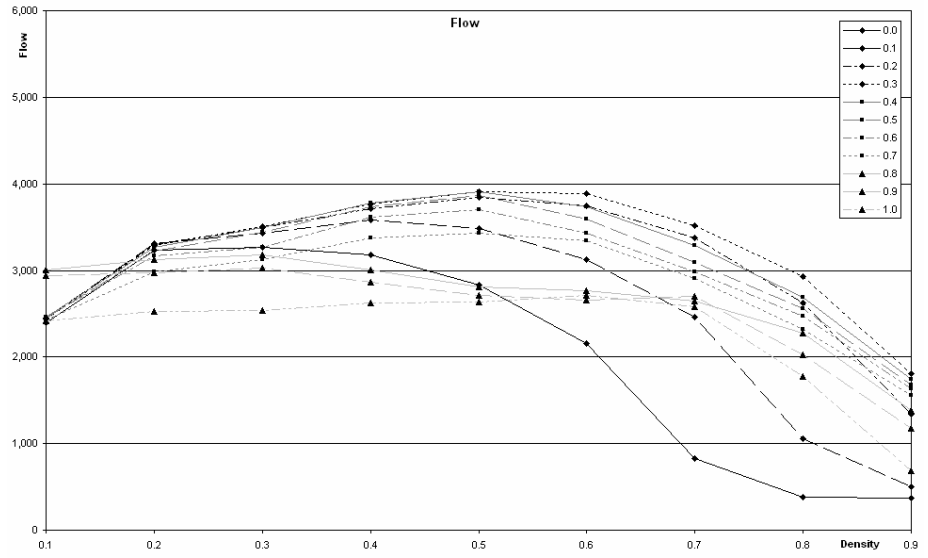

Figure 7.7: Flow of all vehicles in the extended three-road network, which is slightly higher than in the situation with the simple network.

Extended five-road network

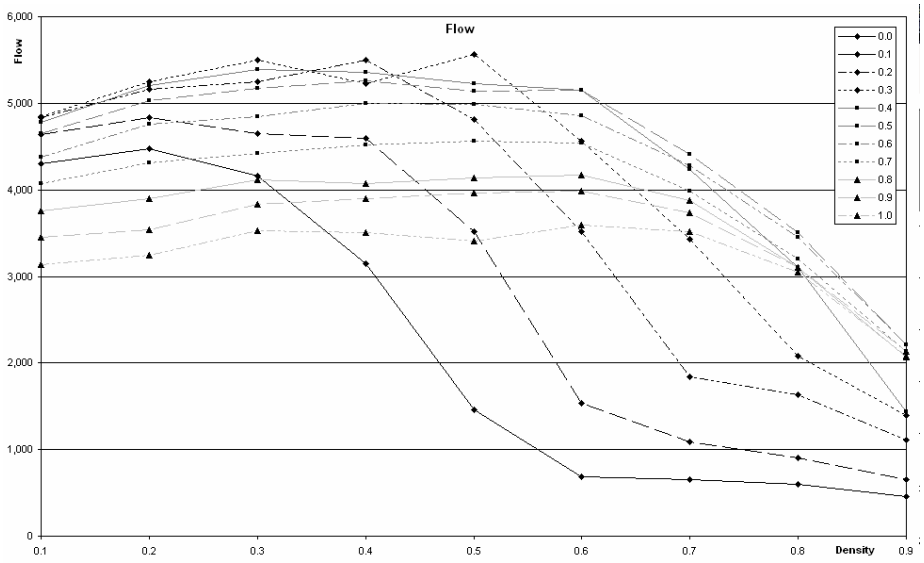

Figure 7.8: Flow of all vehicles in the extended five-road network. The differences are even larger, because of the increase in route possibilities.

7.2 Determining the dependability

To find out the dependability of the model and the networks 99 combinations of δ and ρ are investigated (9 densities, times 11 dynamic ratios). At every combination, the simulation will be run 100 times and the 90% Confidence Interval (CI, see section 6.3) is determined. The difference between the lower bound and the upper bound of this interval is divided by the average of the 100 runs. This way the relative dependability of the simulation with the given parameters is determined, which will be plotted in a chart to show the influence of these parameters δ and ρ on the results. This investigation is done in all three networks, to see whether the layout of the network and the number of roads and interconnects play a part in the degree of confidence of the simulation. The values of these tests can be found in appendix C.

In Figure 7.9 the differences of all tests between the lower and upper bound of the 90% confidence interval is shown. For all lightgrey points in the charts (<30%), the model is very reliable. So, for most of the combinations of (δ, ρ) the outcomes as stated in the previous section are correct. Only on high densities, the flow of the network has a large variance. That is understandable, because the values are low, so a small variance in the output causes large relative differences.
Concluding, the differences are relatively small, so the outcomes of the experiments are reliable enough to make some conclusions on the effects of communication.

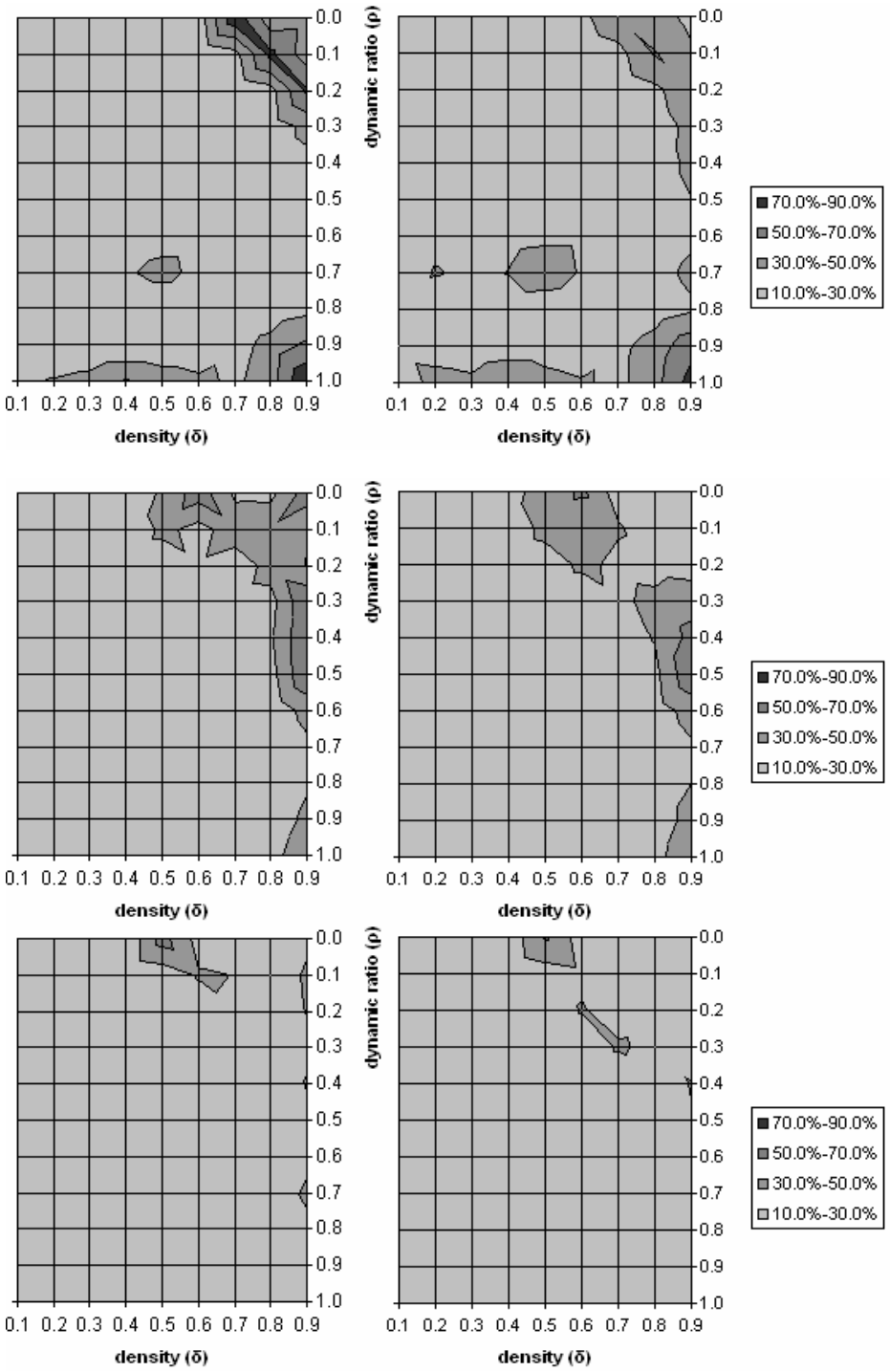

Figure 7.9: Differences in flow (left) and trip time (right) of the three networks, top to bottom: simple three-road, extended three-road and extended five-road network.

8 Conclusions and recommendations

After studying the literature, developing the software and doing the experiment, several conclusions can be made on the effects of adding communication in traffic.

Conclusion

The first and most important conclusion is that the vehicles with communication are always faster than the vehicles without. These differences are at a maximum on high densities, because on low densities there are no traffic jams, so no optimisation can be achieved.

Second conclusion is that not only the communicating vehicles have an advantage over the non-communication ones, but the overall performance also increases. So all drivers will have lower trip times and the network will have a larger throughput.

One of the most striking effects is that the optimal situation does not arise at the situation where everybody uses dynamic routing, but at about 70%. Above this ratio the amount of communication vehicles will be too high, making the alternative routes are getting too full, leading to sub-optimal results.

Summaring these conclusions, adding communication in traffic leads to a decrease in trip times and an increase in the throughput of the network. The capacity of the network will be increased, without the need for increasing the number of roads or lanes.

Recommendations

Of course, this study did not cover averey possible situation and there are always some points of interest which have not been covered. This section suggests a few directions for further studies.

Only one way of communication has been used in the demonstrations. The vehicles can ask the speed and decisions of every other vehicle. In other words, everybody knows everything. A more self-organising solution would be when vehicles can only ask the vehicles in their direct neighbourhood. This form of communication can be implemented in the simulation software created for this study.

Besides a different way of communication and information distribution, the network model can be extended by allowing multi-lane roads. In those situations vehicles are able to overtake, what may lead to other outcomes.

The networks used in these experiments are all “highway-like”. It is also possible to create several “city-like” networks with several begin and end nodes. Using the dense networks, the results may be different. This requires the internal model to be changed. As explained in section 6.1, the experiments are done with a closed network. But city-like networks require an open network. So another recommendation is to implement open network possibilities and test with more dense networks.

In chapter 9 some of the previous experiments done with the closed network are done with open network too. The networks are the same as in the earlier tests, so no real city-like network will be used. This is done to show the influence of connecting the end to the begin node.

9 Recommendation: Open network

The literature about using cellular automata for traffic simulations all use closed networks, with the end node connected to the begin node. But with this approach, city-like simulations can not be made. In this chapter some small and basic tests with open networks will be done, to show the main differences between open and closed networks. This chapter is not intended to give a full view on the exact differences between the closed and open approach.

9.1 Definitions

Most of the definitions can be found in section 6.3, but one of the definitions is slightly different in an open network.

Density (δ)

In the closed networks, the density of a road denoted the "fullness" of it, varying from 0.0 (no vehicles on the road) to 1.0 (every cell of the road is occupied). In these open network experiments it is also used to initialise the roads with a certain amount of vehicles, but after the network has started the density means the chance for each begin-node to spawn a new vehicle. So if the experiment has a density of 0.0, there won't be any vehicles spawning from the starting points. A density of 1.0 means every update a new vehicle is spawned into the network.

9.2 Results of the open network

At first, the trip times of the vehicles are measured. The trip times are plotted against the density (the ratio of new vehicles) in Figure 9.1. Compared to the closed network test (Figure 9.2), the largest difference is the flattening lines in the trip times. In this example, the trip times do not rise as high as in the closed network examples.

Also in the flow of the network big differences are visible. The flow of the open network is shown in Figure 9.3. This chart, if compared to Figure 9.4, lacks the sharp decrease at high densities. There is no decrease in the throughput, because the end nodes are open. In the closed network, vehicles at the end of their route have to wait till the begin node has an empty spot. In the open network they do not have to wait.

This effect visible in both charts can be explained by the fact that the density in the open network never will be as high as in the closed network. Although the ratio of new vehicles joining the network is increased to 1, not all the new vehicles can be placed on the road. This is because the first cell may still be occupied by the last new vehicle. So the density in the network will reach a maximum value, resulting in these flattened outcomes.

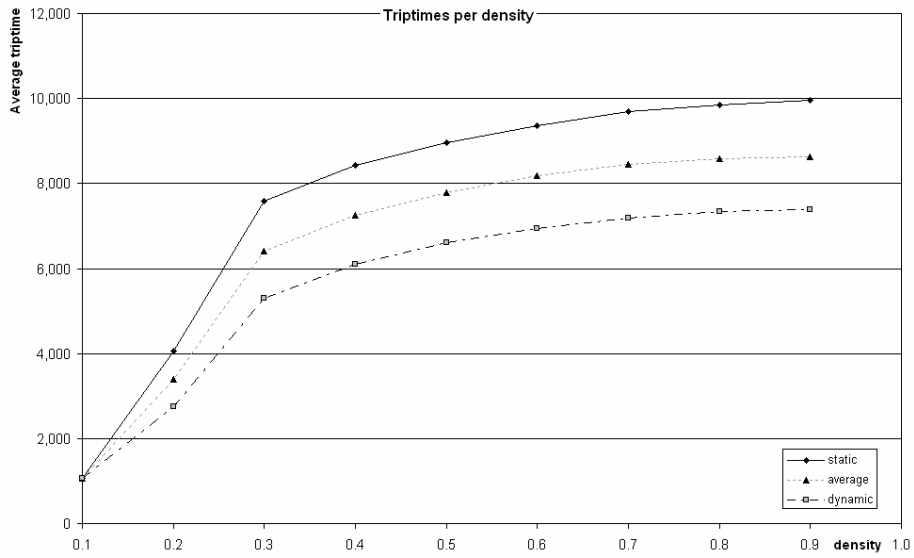

Figure 9.1: Trip times per density in the open three-road extended network

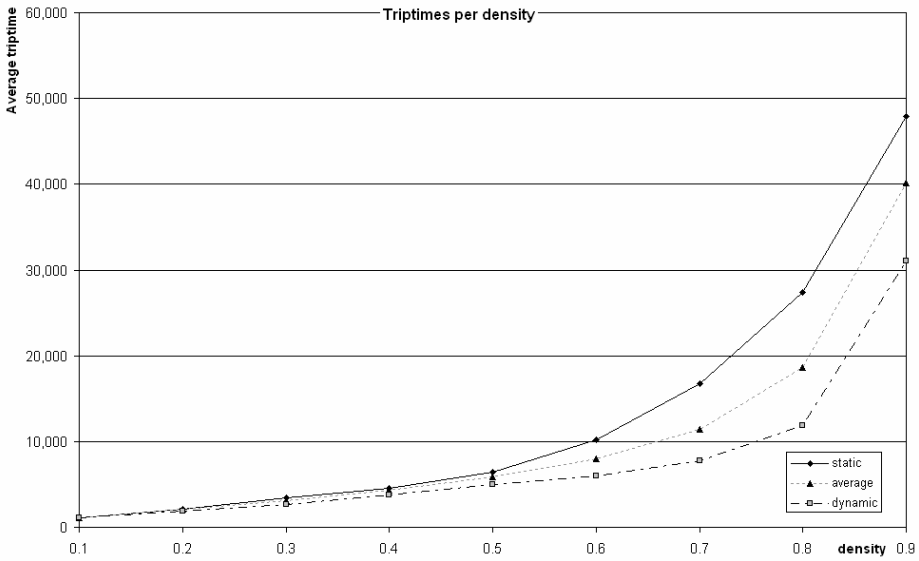

Figure 9.2: Trip times per density in the closed three-road extended network; note the difference in scale between the vertical axis.

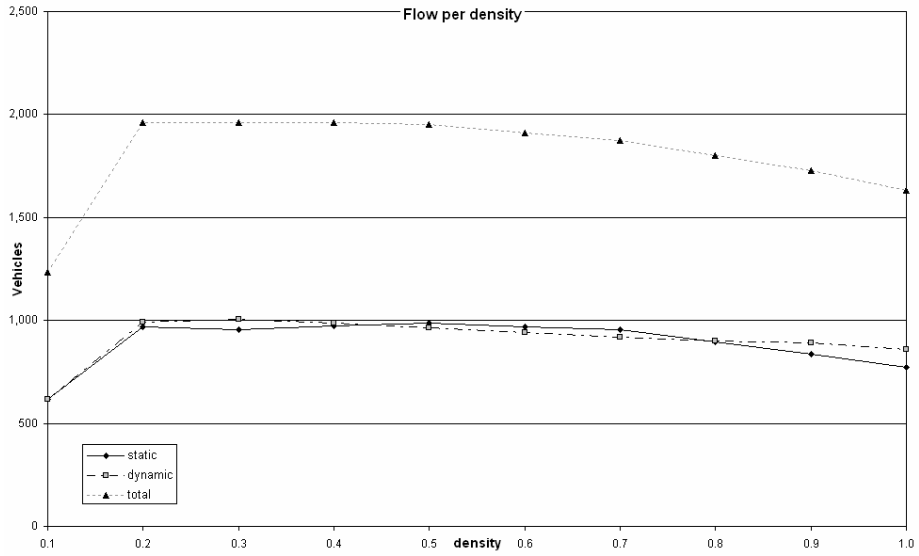

Figure 9.3: Flow per density in the open three-road extended network

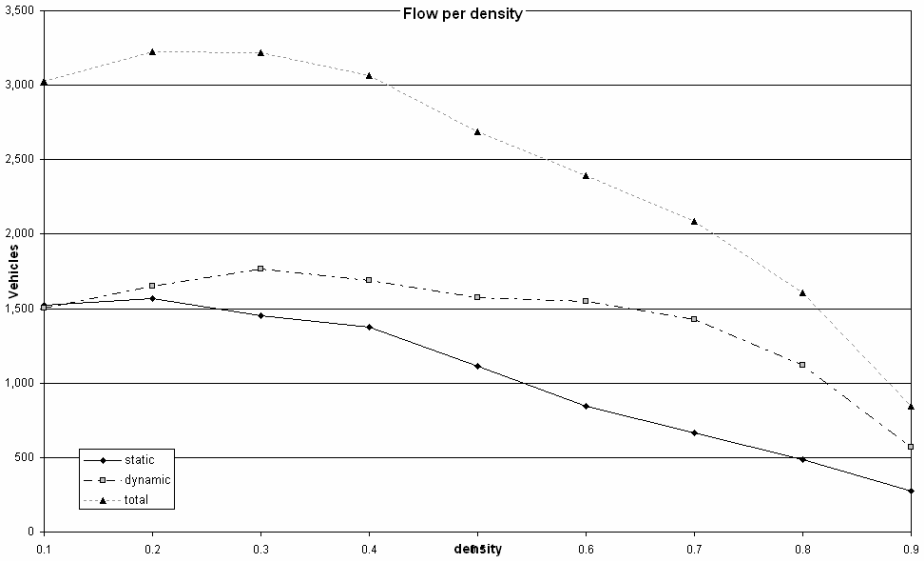

Figure 9.4: Flow per density in the closed three-road extended network

9.3 Conclusion of the open network approach

The effects of dynamical routing and communication in traffic is in closed networks not as striking and clear as in closed ones. However, maybe more dense networks will increase the differences in this open structure. That is something a further study has to find out.

The main conclusion on the differences between closed and open network layouts is that the open networks do not reach the same high densities as closed networks. That leads to charts that level off above a certain ratio of new vehicles spawned, because the density on the roads will stay nearly the same.

References

Documents

[1] Mitchell Resnick; Turtles, termites and traffic jams; The MIT Press; Massachusetts; 0262680939; 1997.

[2] Marco Dorigo, Luca Maria Gambardella; Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem; ftp://ftp.idsia.ch/pub/luca/papers/acs-ec97.pdf; 1997.

[3] Ruud Schoonderwoerd, Owen Holland, Janet Bruten and Leon Rothkrantz; Ant-based load balancing in telecommunications networks; http://dsp.jpl.nasa.gov/members/payman/swarm/schoon96-hpl.pdf; 1996.

[4] Ronald Kroon; Dynamic Vehicle Routing using Ant Based Control; Master's thesis TU Delft; 2002.

[5] Ronald Kroon, Leon Rothkrantz; Dynamic Vehicle Routing using an ABC-algorithm; http://www.fd.cvut.cz/Czech/Events/Sbornik/2003/Doprava_a_Telekomunikace/kroon.pdf; 2003.

[6] Pieter Edelman; Samen Slim; Kijk (monthly magazine), December 2003; pag. 7 - 10; 2003.

[7] Vitorino Ramos, Filipe Almeida; Artificial Ant Colonies in Digital Image Habitats – A Mass Behaviour Effect Study on Pattern Recognition; http://alfa.ist.utl.pt/~cvrm/staff/vramos/Vramos-ANTS00.pdf; 2000.

[8] The Economist; Desirable dust; http://www.economist.com/surveys/displayStory.cfm?Story_id=949030; 2002

[9] D. Braess; Ueber ein paradoxon aus der Verkehrsplanung; http://homepage.ruhr-uni-bochum.de/Dietrich.Braess/paradox.pdf; 1969.

[10] Werkgroep CROW; Langzaam rijden gaat sneller; CROW; 9066283998; 2004.

[11] R. Nachtegaal; Self-organisation in the traffic domain, Literature survey; 2004.

[12] B.S. Kerner and H. Rehborn; Experimental Properties of Phase Transitions in Traffic Flow; Phys. Rev. Let. 79, 4030-4033; 1997.

[13] Debashish Chowdhury, Ludger Santen, Andreas Schadschneider; Statistical Physics of Vehicular Traffic and Some Related Systems; http://xxx.lanl.gov/pdf/cond-mat/0007053; 2000.

[14] R.W. Rothery, in: N. Gartner, C.J. Messner, A.J. Rathi (eds.); Traffic Flow Theory, 2nd ed.; Transportation Research Board (TRB) Special Report 165; 1998.

[15] K. Kaneko (Ed.); Theory and Applications of Coupled Map Lattices; Wiley, New York; 1993.

[16] S. Migowsky, T. Wanschura and P. Rujan; Z. Phys. B 95; p. 211; 1994.
[17] D. Helbing, A. Hennecke, and M. Treiber; Phase diagram of traffic states in the presence of inhomogeneities; Physical Review Letters 82, pag. 4360-4363; 1999.
[18] Kai Nagel, Michael Schreckenberg; A cellular automaton model for freeway traffic; http://www.edpsciences.org/articles/jp1/pdf/1992/12/jp1v2p2221.pdf; 1992.
[19] M. Rickert, K. Nagel, M. Schreckenberg, A.Latour; Two lane traffic simulations using cellular automata; Physika A 231, pag. 534-550; 1995.
[20] P. M. Simon, H. A. Gutowitz; A Cellular Automaton Model for Bi-Directional Traffic; http://www.santafe.edu/~hag/traffic/traffic.html; 1996.
[21] P. M. Simon, H. A. Gutowitz; A Cellular Automaton Model for Bi-Directional Traffic; http://arxiv.org/pdf/cond-mat/9801024; 1998.
[22] Lawrence Gray, David Griffeath; The Ergodic Theory of Traffic Jams; Journal of Statistical Physics, Vol. 105, Nos. 3/4; http://www.math.umn.edu/~gray/pdf/traffic.pdf; 2000.
[23] Preechaya Therakomen; Mouse.class, The Experiments for Exploring Dynamic Behaviors in Urban Places; Master Thesis at the University of Washington; http://depts.washington.edu/dmgftp/publications/pdfs/mouse_class/; 2001.
[24] Ford Motor Company; Intelligent highway revolution: smart vehicles transmit where & how they are; http://media.ford.com/newsroom/release_display_new.cfm?release=17572; 2003.
[25] Dr. Ing. Werner Huber; Fahrzeuggenerierte Daten zur Gewinnung von Verkehrsinformationen; http://www.bmwgroup.com/e/0_0_www_bmwgroup_com/8_science_mobility/8_4_wissenschaft/pdf/8_4_5_Fahrzeuggenerierte_Daten.pdf; 2003.
[26] Yoshiyuki Fujii and Hideki Tsukaoka; Technology to Integrate a Vehicle-Mounted Camera and Image-Processing Unit; http://www.global.mitsubishielectric.com/bu/automotive/advanced_technology/pdf/vol94_tr6.pdf; 2001.
[27] Telematics Forum: GTP (Global Telematics Protocol); GTP Application Protocol: Encoding Specification; 21 March 2003.
[28] Rob Pooley, Perdita Stevens; Using UML. Harlow, Essex: Addison Wesley Longman Limited; 0201360675; 1999.
[29] Jos Warmer, Anneke Kleppe; Praktisch UML; Addison Wesley Longman Nederland B.V.; 9067899372; 1999.
[30] Alistair Cockburn; Basic Use Case Template; HaT technical report TR.96.03a, version 2, Humans and Technology, October 1998; http://alistair.cockburn.us/usecases/uctempla.htm; 1998.

Web sites

[31] Larry Page and Sergey Brin; Google pagerank explained; http://www.google.com/technology/; 2003.

[32] Slashdot; Comments and Moderation; http://slashdot.org/faq/com-mod.shtml; 2003.

[33] Sadayuki Tsukawa; An introduction into Demo 2000: The co-operative driving scenario; http://www.demo2000.gr.jp/index_e.htm; 2000.

[34] GTP-group; Frequently Asked Questions GTP; http://www.telematicsforum.com/document/acp_gats/acrobat/gtp_faq.pdf; 2003.

[35] Barak Naveh e.a.; JGraphT; http://jgrapht.sourceforge.net/; 2003.

[36] Paramics; http://www.paramics-online.com/overview/overview_main.htm; 2004.

A. Use cases

A.1 Use cases of the software user

See Figure 4.1: Use case diagram of the software-user.

Table A.1: Use case Add Node

Name	**Add Node**
Goal	Add a node to the network.
Actor(s)	User.
Precondition(s)	None.
Success End Condition	The Node exists and will be selected.
Failed End Condition	The Node does not exist.
Description	The Node will be added to the network, without connected roads.
Open Issues	None.

Table A.2: Use case Add Road

Name	**Add Road**
Goal	Add a road to the network.
Actor(s)	User.
Precondition(s)	Two nodes have to exist between which the road has to be created. There is no road yet between these nodes in the specified direction (roads can exist in both ways).
Success End Condition	The Road exists and will be selected.
Failed End Condition	The Road does not exist.
Description	The Road will be added to the network, all vehicles will update their network (and maybe their route). If there is a road already between the nodes nothing will be done.
Open Issues	None.

Table A.3: Use case Change Node weight

Name	**Change Node weight.**
Goal	Changes the weight of a Node.
Actor(s)	User.
Precondition(s)	Node is active and an end-Node.
Success End Condition	The weight has changed.
Failed End Condition	Nothing happened.
Description	The weight of the Node will be changed if the Node is an end-Node. Else no change must be possible.
Open Issues	None.

Table A.4: Use case Change Road

Name	**Change Road.**
Goal	Change some property of a Road.
Actor(s)	User.
Precondition(s)	The Road must be active.
Success End Condition	The selected property has to be changed.
Failed End Condition	Nothing has to be changed.
Description	A specified property (like the speed) will be changed. All vehicles have to update their weights.
Open Issues	None.

Table A.5: Use case Move Node

Name	**Move Node.**
Goal	Move a Node to another location.
Actor(s)	User.
Precondition(s)	The Node has to be active.
Success End Condition	The Node is moved.
Failed End Condition	The Node is not moved.
Description	The Node will be moved, after which all connected roads have to be updated according to their new lengths.
Open Issues	None.

Table A.6: Use case Delete Node

Name	**Delete Node.**
Goal	The Node has to be deleted.
Actor(s)	User.
Precondition(s)	The Node is active.
Success End Condition	The Node is removed.
Failed End Condition	The Node is not changed.
Description	The Node will be removed, except when one or more Roads are connected to it. In that case nothing happens.
Open Issues	None.

Table A.7: Use case Delete Road

Name	**Delete Road.**
Goal	The Road will be deleted.
Actor(s)	User.
Precondition(s)	The Road is active.
Success End Condition	The Road is deleted.
Failed End Condition	The Road is still there.
Description	The Road will be removed, all Vehicles on it will be removed and all other Vehicles will have their network updated.
Open Issues	None.

Table A.8: Use case Change Road-weight

Name	**Change Road-weight.**
Goal	Update the weight of a Road
Actor(s)	System.
Precondition(s)	One of the previous use cases was called: Add Road, Change Road or Move Node.
Success End Condition	The weight has been updated.
Failed End Condition	None.
Description	If a Road has been changed, added or moved (by moving one of the connected Nodes), the weight has to be updated.
Open Issues	None.

A.2 Use cases of the vehicles

See Figure 4.2: Use case diagram of the static vehicles and Figure 4.3: Use case diagram of the dynamic vehicles.

Table A.9: Use case Enter network

Name	**Enter network**
Goal	The Vehicle enters the network
Actor(s)	Vehicle.
Precondition(s)	Vehicle must have a begin and end point.
Success End Condition	Vehicle is placed in the network knowing where to go.
Failed End Condition	Vehicle is not placed in the network.
Description	When the Vehicle knows which way to go to its destination, it will be placed on that road. If there is no empty space for a Vehicle, it will wait if it is a closed network and will be discarded in an open network (see section 6.1).
Open Issues	None.

Table A.10: Use case Calculate route

Name	**Calculate route.**
Goal	Calculates the optimal route to a specified end node, returning the first road of the route.
Actor(s)	Use case *Enter network* and *Reach Node* (for dynamical routing vehicles only).
Precondition(s)	Current node and end node have to be known.
Success End Condition	Returns the first road of the calculated route.
Failed End Condition	Returns null to tell there is no route to the end node.
Description	Uses the internal weights of each road to calculate the route with the lowest weight to the end node.
Open Issues	None.

Table A.11: Use case Check for dead-end street

Name	**Check for dead-end street.**
Goal	Check whether there is a route from here to the end node.
Actor(s)	Use case *Calculate route*.
Precondition(s)	Current node and end node have to be known.
Success End Condition	Boolean whether there is a route from here to the end node.
Failed End Condition	None.
Description	If there is no route from the current node to the end node, there is no use for the vehicle to go to a next road. The outcome of this case will be used in *Calculate route*.
Open Issues	None.

Table A.12: Use case Ask speed of roads

Name	**Ask speed of Roads.**
Goal	Get to know the speed (and weight) of a road.
Actor(s)	Use case *Calculate route*.
Precondition(s)	The road exists in the network.
Success End Condition	The speed of the road will be returned.
Failed End Condition	Nothing will be returned.
Description	There are two possibilities; the road can return its maximum speed (for the statically routing vehicles) or the current average speed of all vehicles on this road (for the dynamically routing vehicles).
Open Issues	None.

Table A.13: Use case Reach Node

Name	**Reach Node.**
Goal	Check next action of the vehicle (next road or destination reached).
Actor(s)	Vehicle.
Precondition(s)	Vehicle has a destination and is at the end of a road.
Success End Condition	Vehicle knows next road to take or it is at its destination and will be discarded.
Failed End Condition	Stay at the last cell of the road.
Description	If a Vehicle reaches a Node, it will check whether it reached its end Node, or determine the next road to take. Dynamic routing Vehicles will recalculate their route to take the latest situation into account.
Open Issues	None.

Table A.14: Use case Check for destination

Name	**Check for destination.**
Goal	Know whether the current Node is the destination of this Vehicle.
Actor(s)	Use case *Reach Node*.
Precondition(s)	Vehicle has an end Node and is at a Node right now.
Success End Condition	Boolean whether or not it is at its destination.
Failed End Condition	Error.
Description	If a Vehicle reaches its destination, it has to be discarded. Otherwise it has to travel to the next node of its route.
Open Issues	None.

Table A.15: Use case Stop

Name	**Stop.**
Goal	The vehicle has stopped and will be removed from the network.
Actor(s)	Use case *Check for dead-end street* and *Check destination*.
Precondition(s)	Vehicle reaches a node.
Success End Condition	Vehicle will be discarded from the network.
Failed End Condition	None.
Description	Vehicle can be at its destination, or at a Node from which there is no possibility to get to the end Node. In both cases the Vehicle has to be removed from the network.
Open Issues	None.

B. XML network example

This is an example of an XML file containing a very simple road network. It consists of two nodes and one road between them.

```
<?xml version="1.0" encoding="UTF-8"?>
<network>
<nodes>
<node>
<id>0</id>
<p>10,10</p>
<color>6513664</color>
<newVehicleRatio>0.02</newVehicleRatio>
<weight>1.0</weight>
</node>
<node>
<id>1</id>
<p>890,490</p>
<color>16645376</color>
<newVehicleRatio>0.1</newVehicleRatio>
<weight>1.0</weight>
</node>
</nodes>
<roads>
<road>
<beginNode>0</beginNode>
<endNode>1</endNode>
<speed>160</speed>
</road>
</roads>
</network>
```

C. Tables

In this appendix, the tables with all numbers from the tests will be shown.

C.1 Results

The following tables contain all outcomes of the experiments with the different networks.

C.1.1 Trip times

Simple three-road network

Table C.1: Trip times in the simple three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	978	1,453	2,152	2,953	4,100	6,289	15,081	34,232	52,424
0.1	1,917	2,698	3,974	4,861	5,874	7,350	10,406	23,644	56,195
0.2	1,914	2,818	3,998	4,944	5,859	6,975	8,863	13,223	28,483
0.3	1,912	2,844	4,032	4,957	5,964	7,128	9,127	12,499	23,093
0.4	1,915	2,868	4,038	4,961	6,056	7,534	9,923	13,736	24,759
0.5	1,912	2,905	4,080	5,043	6,185	7,969	10,736	14,663	25,610
0.6	1,918	2,948	4,281	5,250	6,489	8,323	11,156	15,429	26,904
0.7	1,914	3,149	4,442	5,612	7,088	8,533	11,328	16,752	29,276
0.8	2,237	4,328	6,470	8,937	11,899	14,595	17,732	23,984	46,159
0.9	2,282	4,517	6,688	9,300	12,239	15,105	12,253	19,179	45,206
1.0	962	1,837	2,756	3,616	4,412	5,188	6,387	10,671	32,592

Extended three-road network

Table C.2: Trip times in the extended three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	1,148	2,169	3,491	6,029	16,640	34,621	63,832	75,646	92,377
0.1	1,109	2,059	3,234	4,305	7,456	12,766	25,877	43,048	65,252
0.2	1,088	2,002	3,092	4,045	5,406	8,731	13,316	30,281	62,995
0.3	1,092	1,991	2,994	4,077	4,996	7,040	11,056	19,631	62,230
0.4	1,087	1,994	2,899	4,013	5,352	6,758	9,599	16,984	51,494
0.5	1,100	1,994	2,930	3,951	5,538	6,970	9,066	14,132	36,127
0.6	1,108	2,018	2,983	4,120	5,713	7,027	8,311	11,778	24,867
0.7	1,110	2,082	3,067	4,283	5,858	7,145	8,543	11,366	22,359
0.8	1,128	2,166	3,193	4,494	6,057	7,296	8,852	11,939	23,033
0.9	1,156	2,289	3,351	4,701	6,252	7,643	9,392	12,635	24,709
1.0	1,191	2,463	3,548	5,078	6,628	8,048	10,127	14,098	28,583

Extended five-road network

Table C.3: Trip times in the extended five-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	1,392	2,659	4,355	7,472	19,145	49,149	63,871	78,633	93,874
0.1	1,286	2,475	3,774	5,164	8,236	20,269	37,661	51,415	73,508
0.2	1,237	2,325	3,386	4,348	6,086	9,896	20,860	29,319	44,709
0.3	1,233	2,293	3,240	4,524	5,363	7,783	12,076	21,983	37,727
0.4	1,252	2,318	3,343	4,462	5,677	6,981	9,891	15,683	37,377
0.5	1,291	2,403	3,477	4,530	5,775	6,916	9,403	13,690	25,235
0.6	1,365	2,526	3,707	4,779	6,005	7,367	9,758	14,193	24,414
0.7	1,477	2,759	4,040	5,244	6,531	7,838	10,369	14,965	25,084
0.8	1,587	3,040	4,357	5,796	7,120	8,533	10,699	15,257	25,502
0.9	1,725	3,342	4,667	6,078	7,491	8,982	11,115	15,369	25,181
1.0	1,866	3,624	5,047	6,831	8,692	10,001	11,876	15,670	25,819

C.1.2 Flow

Simple three-road network

Table C.4: Flow of the simple three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	2,399	3,229	3,268	3,175	2,826	2,158	830	379	373
0.1	2,446	3,305	3,434	3,583	3,490	3,123	2,456	1,055	500
0.2	2,446	3,302	3,498	3,718	3,840	3,745	3,373	2,626	1,343
0.3	2,443	3,291	3,511	3,769	3,914	3,889	3,522	2,924	1,812
0.4	2,460	3,261	3,498	3,775	3,912	3,734	3,288	2,694	1,739
0.5	2,450	3,222	3,440	3,738	3,861	3,598	3,089	2,557	1,675
0.6	2,465	3,164	3,270	3,613	3,706	3,434	2,988	2,468	1,629
0.7	2,433	2,987	3,129	3,377	3,434	3,341	2,906	2,323	1,552
0.8	3,008	3,127	3,182	3,010	2,810	2,763	2,650	2,272	1,380
0.9	2,941	2,974	3,024	2,860	2,714	2,656	2,697	2,024	1,172
1.0	2,422	2,528	2,541	2,623	2,632	2,714	2,578	1,776	684

Extended three-road network

Table C.5: Flow of the extended three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	2,905	3,066	2,946	2,292	825	523	353	337	220
0.1	3,041	3,278	3,133	3,197	2,222	1,378	817	599	408
0.2	3,102	3,380	3,223	3,319	3,113	2,017	1,616	858	449
0.3	3,125	3,418	3,333	3,190	3,320	2,834	2,025	1,337	462
0.4	3,103	3,427	3,466	3,259	3,140	2,978	2,424	1,597	570
0.5	3,094	3,412	3,471	3,419	3,033	2,887	2,607	1,897	868
0.6	3,070	3,363	3,412	3,304	2,943	2,884	2,867	2,314	1,249
0.7	3,039	3,280	3,317	3,153	2,893	2,853	2,774	2,387	1,385
0.8	3,004	3,124	3,182	3,018	2,804	2,782	2,650	2,255	1,362
0.9	2,948	2,964	3,038	2,885	2,712	2,664	2,517	2,131	1,254
1.0	2,856	2,758	2,853	2,666	2,563	2,507	2,330	1,922	1,063

Extended five-road network

Table C.6: Flow of the extended five-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	4,299	4,473	4,159	3,143	1,464	689	657	597	456
0.1	4,637	4,836	4,652	4,595	3,517	1,539	1,084	903	648
0.2	4,831	5,161	5,252	5,498	4,818	3,518	1,840	1,636	1,107
0.3	4,850	5,252	5,501	5,226	5,564	4,561	3,433	2,081	1,394
0.4	4,783	5,206	5,391	5,355	5,231	5,145	4,232	3,099	1,441
0.5	4,646	5,030	5,176	5,258	5,138	5,155	4,413	3,509	2,214
0.6	4,382	4,762	4,848	4,995	4,986	4,854	4,283	3,448	2,206
0.7	4,072	4,317	4,416	4,524	4,567	4,544	3,984	3,204	2,131
0.8	3,759	3,900	4,111	4,070	4,134	4,170	3,871	3,106	2,073
0.9	3,447	3,536	3,837	3,904	3,962	3,989	3,737	3,101	2,134
1.0	3,136	3,246	3,526	3,509	3,412	3,590	3,521	3,051	2,081

C.2 Confidence Interval

These tables contain all relative differences between the lower bound and the upper bound of the 90% Confidence Interval of the tests. The columns are the densities (δ) and the rows are the ratios of dynamic routing (ρ).

C.2.1 Trip times

Simple three-road network

Table C.7: CI bounds of the trip times in the simple three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	7.7%	17.9%	12.0%	16.4%	16.8%	22.9%	48.7%	38.4%	24.2%
0.1	5.4%	21.3%	9.9%	8.7%	10.7%	12.2%	22.2%	51.4%	33.5%
0.2	5.2%	16.1%	11.0%	8.7%	8.0%	9.7%	15.0%	24.9%	45.7%
0.3	4.7%	18.8%	10.0%	8.6%	9.4%	10.1%	16.7%	20.6%	34.9%
0.4	4.5%	19.2%	12.4%	9.8%	13.1%	14.2%	20.5%	17.4%	35.9%
0.5	4.7%	16.7%	12.8%	12.6%	15.6%	19.3%	18.7%	16.2%	29.4%
0.6	5.2%	25.2%	19.9%	21.5%	24.3%	25.1%	21.2%	20.0%	26.5%
0.7	5.0%	31.4%	25.5%	30.6%	45.2%	27.8%	20.8%	24.2%	33.7%
0.8	19.0%	15.6%	13.2%	15.4%	14.0%	11.4%	13.4%	19.7%	27.1%
0.9	20.2%	17.0%	14.2%	18.5%	17.9%	13.5%	24.8%	37.8%	62.4%
1.0	5.7%	40.0%	36.0%	49.6%	39.0%	32.9%	24.4%	42.3%	77.3%

Extended three-road network

Table C.8: CI bounds of the trip times in the extended three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	18.2%	19.4%	18.7%	26.2%	34.9%	53.9%	19.7%	12.5%	9.3%
0.1	10.4%	17.3%	13.1%	12.6%	37.5%	31.1%	31.9%	21.9%	19.5%
0.2	12.7%	13.9%	12.0%	12.9%	19.0%	34.2%	26.0%	23.5%	20.9%
0.3	13.8%	12.4%	16.0%	14.8%	13.5%	15.7%	26.8%	33.9%	42.2%
0.4	14.5%	14.8%	13.4%	14.6%	11.4%	11.6%	15.5%	31.2%	57.1%
0.5	17.4%	15.2%	10.3%	16.9%	12.4%	11.7%	15.8%	23.7%	67.4%
0.6	16.2%	19.4%	10.5%	12.6%	14.7%	10.0%	13.1%	19.3%	37.6%
0.7	15.3%	16.4%	13.1%	13.9%	12.9%	10.2%	13.1%	16.6%	27.3%
0.8	16.0%	16.0%	13.9%	17.1%	12.9%	10.4%	14.8%	17.1%	29.9%
0.9	17.4%	16.2%	15.1%	16.6%	14.4%	12.9%	16.9%	17.2%	38.8%
1.0	23.0%	17.9%	15.9%	15.6%	13.0%	12.3%	14.1%	24.9%	40.8%

Extended five-road network

Table C.9: CI bounds of the trip times in the extended five-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	14.7%	12.4%	13.7%	16.9%	52.2%	19.7%	12.2%	9.2%	8.8%
0.1	10.6%	9.8%	9.4%	13.2%	20.8%	25.2%	16.0%	12.7%	15.8%
0.2	12.2%	9.6%	12.7%	10.0%	14.7%	31.6%	21.8%	22.9%	12.3%
0.3	18.5%	10.0%	10.5%	11.9%	12.0%	13.9%	32.5%	24.6%	15.9%
0.4	14.2%	11.9%	10.7%	12.1%	13.2%	10.2%	13.2%	20.1%	31.1%
0.5	21.3%	13.2%	11.2%	9.5%	8.2%	13.9%	12.3%	14.9%	28.5%
0.6	25.3%	14.0%	10.2%	10.5%	11.0%	14.4%	18.2%	23.5%	21.2%
0.7	26.5%	17.3%	17.4%	16.5%	16.8%	16.6%	18.2%	21.0%	27.2%
0.8	25.5%	17.6%	18.6%	19.8%	20.2%	19.5%	15.9%	20.9%	21.3%
0.9	21.6%	15.2%	19.1%	18.4%	18.3%	15.0%	14.2%	21.7%	21.2%
1.0	21.8%	14.2%	18.9%	18.0%	17.3%	16.9%	15.1%	15.8%	25.8%

C.2.2 Flow

Simple three-road network

Table C.10: CI bounds of the flow of the simple three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	18.4%	4.1%	2.3%	3.8%	9.3%	17.3%	84.2%	36.5%	39.7%
0.1	23.1%	2.5%	3.1%	3.8%	7.3%	11.3%	21.8%	74.8%	39.4%
0.2	24.5%	2.2%	3.9%	3.8%	3.5%	5.5%	12.1%	22.0%	72.9%
0.3	21.5%	3.1%	4.4%	3.5%	2.9%	5.2%	12.3%	18.9%	34.8%
0.4	20.4%	3.4%	4.4%	3.8%	4.8%	11.5%	16.0%	19.0%	25.4%
0.5	20.9%	6.5%	8.0%	5.3%	7.0%	15.2%	16.3%	17.1%	24.4%
0.6	22.3%	10.2%	16.3%	16.3%	19.6%	20.5%	20.3%	20.4%	22.4%
0.7	22.6%	20.5%	21.2%	26.7%	37.1%	23.5%	18.4%	22.8%	27.9%
0.8	6.2%	7.1%	9.3%	8.0%	9.2%	7.4%	13.6%	17.0%	24.5%
0.9	5.9%	7.4%	10.0%	11.0%	11.0%	9.5%	22.3%	36.6%	53.7%
1.0	22.0%	32.4%	36.4%	51.9%	42.3%	35.9%	25.6%	40.8%	89.7%

Extended three-road network

Table C.11: CI bounds of the flow of the extended three-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	6.9%	4.6%	5.1%	19.5%	32.2%	61.2%	25.3%	23.5%	60.7%
0.1	11.1%	3.8%	5.4%	7.8%	36.3%	21.2%	43.3%	47.3%	30.9%
0.2	10.9%	2.7%	6.3%	11.9%	12.5%	26.0%	15.2%	38.0%	29.3%
0.3	7.8%	2.6%	6.8%	12.6%	8.5%	11.1%	22.1%	23.3%	66.6%
0.4	9.3%	2.6%	5.7%	11.0%	7.6%	9.5%	11.6%	27.1%	67.8%
0.5	7.5%	3.4%	4.2%	8.8%	7.1%	6.7%	14.5%	22.2%	68.8%
0.6	7.5%	4.1%	5.1%	8.0%	6.9%	6.0%	9.5%	15.3%	36.9%
0.7	7.0%	5.4%	7.3%	8.7%	7.8%	8.4%	12.1%	12.9%	25.7%
0.8	6.9%	6.6%	10.2%	10.2%	8.9%	8.5%	13.9%	16.3%	27.6%
0.9	7.6%	6.3%	10.8%	9.4%	10.4%	9.7%	15.4%	19.0%	34.1%
1.0	8.4%	7.3%	10.9%	11.2%	11.4%	10.8%	12.3%	25.8%	38.4%

Extended five-road network

Table C.12: CI bounds of the flow of the extended five-road network

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	4.0%	3.9%	6.3%	11.8%	58.7%	21.4%	21.3%	20.0%	28.8%
0.1	5.5%	4.7%	11.0%	10.3%	17.4%	31.8%	29.7%	26.3%	30.9%
0.2	3.0%	3.3%	5.9%	11.9%	9.3%	20.9%	28.0%	21.9%	30.4%
0.3	3.1%	3.8%	6.1%	6.3%	8.8%	12.5%	22.4%	20.0%	25.8%
0.4	4.4%	4.1%	4.7%	5.4%	13.3%	7.4%	9.8%	16.3%	31.2%
0.5	7.7%	5.8%	5.3%	5.6%	6.3%	10.0%	10.4%	14.6%	25.6%
0.6	11.8%	8.1%	9.2%	9.6%	9.0%	14.8%	17.2%	16.5%	26.0%
0.7	15.1%	12.9%	14.4%	14.9%	17.9%	18.1%	21.0%	21.4%	32.2%
0.8	18.3%	12.8%	17.2%	18.7%	21.1%	22.0%	17.3%	19.0%	26.7%
0.9	11.3%	10.2%	17.5%	18.9%	16.3%	17.2%	14.3%	19.0%	27.5%
1.0	12.0%	10.0%	20.3%	15.6%	16.9%	16.8%	17.6%	16.6%	30.0%