UPTEC STS07 003
Examensarbete 20 p
Februari 2007

Automatic Feedback Control Applied to Microscopically Simulated Traffic

The potential of route guidance in the Berlin traffic network

Carl Rommel

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Automatic Feedback Control Applied to Microscopically Simulated Traffic

Carl Rommel

In this master thesis, the traffic in a microscopic traffic simulation of Berlin (MATSim) is controlled by route guidance. The guidance is directed by automatic feedback control and transmitted to the simulated vehicles via in-car COOPERS-devices. The aim of the control is to achieve and maintain a Nash equilibrium on two alternative routes between a common origin and a common destination in the traffic network. Since no explicit model of the controlled traffic simulation is available, only algorithmically simple control strategies such as bang-bang control, proportional control (P-control) and proportional and integral control (PI-control) are applied. The control is applied in two different network settings: a topologically simple test network and a reduced version of the full Berlin network. The main result of the simulations is that well tuned automatic feedback controllers improve the traffic conditions considerably in a scenario where an accident is generated on one of the controlled routes. Generally speaking, P- and PI-control are better at handling the scenario they are tuned to handle, whereas the bang-bang control is more robust against parameter variations. Especially the P-controller's ability to handle changes in the compliance rate is strikingly poor. Given the resemblance between microscopic simulations and reality, it is likely that the route guidance control, that is shown to be successful in the simulations, would work acceptably also in the real-world. In the concluding chapter of the thesis, a broad outline for how such a real-world implementation could be materialized is presented.

Handledare: Kai Nagel
Ämnesgranskare: Bengt Carlsson
Examinator: Elísabet Andrésdóttir
ISSN: 1650-8319, UPTEC STS07 003
Sponsor: COOPERS

Populärvetenskaplig sammanfattning

Igenkorkade gatunät är ett stort problem i de allra av världens stora och medelstora städer. Ibland beror dessa trafikproblem på att vägnätets kapacitet är för liten, men minst lika ofta uppstår köer eftersom trafikflödet är ojämnt fördelat mellan olika alternativa vägar. I en sådan situation finns det en potentiell möjlighet att förbättra den sammantagna trafiksituationen genom att dirigera om trafikanter från överbelastade till mindre belastade vägar. Belastningen på olika alternativa vägar tenderar dock att se mycket olika ut från en dag till en annan och det är därför mycket svårt att på förhand veta vilken väg som kommer att ha hög respektive låg belastning. För att en omdirigering av trafiken ska bli framgångsrik, krävs därför att den aktuella trafiksituationen tas i bekantande då vägvisningen ges till trafikanterna; vägvisningen måste med andra ord vara dynamisk.

Inom ramen för detta examensarbete dirigeras trafiken i en mikroskopisk JAVA-programmerad trafiksimulering av Berlin, med hjälp av dynamisk vägvisning. Syftet med vägvisningen är att uppnå en Nash-jämvikt, dvs. att ingen trafikant väljer en väg som inte är den snabbaste möjliga. Vägvisningen ges till de simulerade trafikanterna via en elektronisk s.k. COOPERS-vägvisare som alla trafikanter antas ha installerad i sin bil. Sättet att dirigera bilarna bestäms på reglerteknisk väg med hjälp av en återkopplande regulator. I en sådan regulator beräknas vägvisningen utifrån den nuvarande trafiksituationen och en förbestämd regleralgoritm Eftersom det inte går att få fram en explicit modell av den typ av simulering som regleras, används enbart mycket enkla reglerstrategier, så som bang-bang-reglering, proportionell reglering (P-reglering) och proportionell och integrerande reglering (PI-reglering).

Vägvisningen implementeras dels i ett litet testnätverk och dels i ett större Berlinnätverk. Simuleringarna visar med tydlighet att regleringen förbättrar trafiksituationen avsevärt i bägge nätverken, då en trafikolycka som innebär långa köer genereras på en av två alternativa vägar. P- och PI-reglering fungerar generellt sett bäst, men det finns tecken på att bang-bang-reglering har en större förmåga att klara av situationer då trafiken beter sig annorlunda än förväntat (t.ex. då olyckan är större än väntat eller då färre trafikanter än väntat följer vägvisningen). Valet av reglerstrategi bör därför göras specifikt i varje enskilt fall. Då den mikroskopiska simuleringen av Berlins trafik har visat sig vara en mycket bra representation av de verkliga trafikflödena i Berlin, finns det anledning att tro att den reglering som fungerar bra i simuleringen också borde kunna förbättra trafiksituationen också i verkligheten. I examensarbetets avslutning finns därför ett förslag på en arbetsgång för hur man skulle kunna överföra de regulatorer som utvecklats i simuleringen till den verkliga stadsmiljön.

Table of Contents

1 Introduction

1.1 The potential of route guidance

Congested traffic networks are of major concern in almost all large cities in the world today. Seeing that traffic congestion is a result of over-utilization, traditional ways of coping with the problem have either been focused on expanding the capacity of the system, i.e. by construction of more and bigger roads, or on reducing the demand, e.g. by means of congestion charging or higher fuel taxes. Congested traffic networks do however not necessarily imply that the capacity of the entire system is insufficient for the demand. Typically, just a fraction of the streets in a city is congested at a given time instance, whereas the traffic load on other streets is comparably low. In such a situation, suitable guidance of the traffic around the problematic areas might improve the traffic conditions considerably. The adoption of a guidance strategy has a potential of solving the congestion problems smoothly, avoiding both expensive road project and the often politically problematic congestion charges and tax increases. Moreover, it is a cheap strategy to implement, since the only technology needed is some kind of infrastructure for transferring the guidance information to the travelers.[1]

One fundamental assumption in traffic engineering and traffic modeling is however that people in a normal day scenario tend to choose the best route available to them also without guidance.[2] No traveler could therefore gain anything from choosing an alternative route. In such a situation, the traffic system is said to have reached a Nash-Equilibrium[3] and no guidance thus seem to be needed. The situation is however only a Nash-equilibrium from a static point of view, i.e. the route chosen by each traveler is the best given the information about the other travelers' behavior that he or she had *prior to* the trip. In the short run, the traffic in a city typically fluctuates strongly and the system is thus not in a Nash-equilibrium at every given time instance. Hence it follows, that it is indeed possible to improve the traffic conditions using guidance, given that this guidance is dynamic and that it takes the present traffic situation into consideration.[4]

---

[1] Papageorgiou, M., "Traffic Control" in Hall, R. W., ed. *Handbook of Transportation Science* 2nd edition (New York, 2003). p 243-244.

[2] Almost all traffic assignment is based on the validity of this assumption. A discussion about the implications of the assumption is found in Nagel, K., Rickert, M., Simon, K. M., Pieck, M., "The dynamics of iterated transportation simulation" (2000), in *Preprints of the TRISTAN-3 Conference, San Juan, Puerto Rico, 1998*, p 2.

[3] A Nash equilibrium is a general, game-theoretical concept that is not traffic specific. In a game with two or more agents, a Nash equilibrium is reached when "no player has anything to gain by changing only his or her own strategy unilaterally". "Nash equilibrium", Wikipedia; The Free Encyclopaedia http://en.wikipedia.org/wiki/Nash_equilibrium (2007-01-08). The concept will be discussed more thoroughly in Chapter 1.4.

[4] The assumption that a traffic system naturally tends towards a Nash equilibrium is in other words rather an approximation that makes modelling possible than a theorem that is always valid in the traffic system.

1.2 Route guidance as applied control theory

1.2.1 Open loop control

The problem of how to guide the traffic through the traffic network at a given time instance using information about the overall traffic conditions is clearly a control problem. Seen from this perspective, the traffic condition is an output signal of a dynamical system that one wants to control desirably by means of giving the system the right input signals, i.e. route guidance.[5] Given such a control problem, there are plenty of strategies available for how to guide the traffic through the network in the best possible way. Looking at published research articles, the most commonly used approaches rely on open loop control in one way or another. [6] Such approaches are based on an often complex mathematical model of the traffic system that describes the relation between guidance (input) and traffic conditions (output). The guidance one uses is then determined by optimizing the output with respect to the input over a given time frame. Since the guidance can typically be changed several times over the optimizing time frame, the optimization problem often ends up being highly dimensional and thus computationally costly.

Assuming that the optimizing algorithm is capable of finding an optimal input, the advantage of controlling by means of open loop control is that the input obtained really is the very best way of controlling the model. The obvious disadvantage is that, due to inevitable disturbances and model errors, the optimal input to the model rarely generates an optimal output also in the real system. Contrary, the input given is likely to generate an output that, although it might be optimal initially, quickly deviates considerably from this optimality. In a highly disturbed system controlled over a long time frame, it thus tends to be difficult to produce a good output by means of open loop control.[7]

1.2.2 Feedback control

The problems with open loop control can be reduced or even eliminated if one takes the development of the real system into consideration when the input is determined. Strategies utilizing this principle are called *feedback* control strategies. One way of using feedback that shows great resemblance with many open loop strategies, is called *Rolling horizon* or *Model Predictive Control* (MPC). When this strategy is used the input is re-optimized at certain time intervals, taking the true development of the

---

[5] A formal definition of the control problem (in Swedish) is given in Glad, T. & Ljung, L., *Reglerteknik; Grundläggande teori* 2nd edition (Lund, 1989), p 10.

[6] A few examples of articles in which traffic is controlled by an open loop approach are: Kotsialos, A., Papageorgiou, M., Mangeas, M., Haj-Salem, H., "Coordinated and integrated control of motorway networks via non-linear optimal control" *Transportation Research Part C 10* (2002), pp 65-84, Bhouri, N., Papageorgiou, M., Blosseville, J. M., "Optimal control of traffic flow on periurban ringways with application to the Boulevard Périphérique in Paris", in *Preprints of the 11th IFAC World Congress, vol 10* (1990), pp 236-243 and Messmer, A. & Papageorgiou, M., "Route diversion control in motorway networks via nonlinear optimisation" *IEEE Transactions and Control Systems Technology 3*, pp 144-154.

[7] Glad, T. & Ljung, L., *Reglerteori; Flervariabla och olinjära metoder* 2nd edition (Lund, 2003), p 451-452.

system into consideration. Compensations for disturbances and model errors are thus carried out continuously. [8]

The control signal of a MPC-controller can however not be updated arbitrarily frequent, since a costly optimization algorithm has to be run every time round. Nevertheless, there are many feedback control strategies that do not rely on any costly optimization calculations at all. Instead, these strategies perform comparably simple calculations (typically vector-matrix-multiplications) in every time step, which generate an input suitable for the present output of the system. The input to the system can therefore be changed more frequently than when a control approach such as MPC is utilized. This kind of control strategies are often denoted *automatic feedback control strategies*. Taking the flexibility of such strategies into consideration, it seems reasonable to assume automatic feedback control to be superior to open loop control approaches when it comes to route guidance through traffic networks. The reason for that is that a traffic system is highly complex and incorporates a high degree of human behavior that is very difficult to model correctly. The interdependence with society also makes it a system with a large number of immeasurable disturbances.[9] In order to control such a system successfully, the robustness against model perturbations and disturbances achieved by feedback based control strategies seems more important than the strict optimality given by optimized open loop control. Furthermore, some of the most basic automatic feedback control strategies also have the advantage that they can be implemented even when an explicit model of the traffic system is lacking. Such "blind" control usually results in surprisingly good output performance of the system, although a differential equation model of the system usually simplifies the process of tuning the controller correctly.[10]

Promising results, using automatic feedback control in traffic engineering have been presented in several articles written by the professor at the Dynamical Systems and Simulation Laboratory, Technical University of Crete, Markos Papageorgiou.[11] The way of controlling the traffic, most commonly used by Papageorgiou is by installation of variable message signs (VMS) in one or a few important intersections in the traffic network. These signs either give the road-users information about expected travel times on some routes downstream of the sign. Alternatively, the signs suggest the traveler

---

[8] Glad & Ljung (2003), pp 423-432.

[9] Example of such disturbances could be: sport events, people that want to go to the beach since the sun is shining, a demonstration etc.

[10] A general description of the idea behind automatic feedback control is found in Glad & Ljung (1989), p 11-12.

[11] A general discussion about feedback control in traffic system is found in Papageorgiou (2003), pp 255-270. A more detailed and mathematical focused article is: Messmer, A. & Papageorgiou, M., "Automatic control methods applied to freeway network traffic" *Automata Vol. 30* (1994)*, No. 4*, pp 691-712. An interesting article where a method based on predictive control is applied is: Wang, Y., Papageorgiou, M., Messmer, A., "A predictive feedback routing control strategy for freeway network traffic" for the Transportation Research Board. 82nd Annual Meeting, January 12-16, 2003, Washington, D. C.

which route to take towards a certain destination. In that case, the signs are usually denoted "Variable destination signs" (VDS).[12]

Papageorgiou and his research group's main focus has been on running two structurally similar simulation tools called METACOR and METANET. The core of both these simulation tools is a nonlinear second order differential equation represented by a state-space model. In this model, the states consist of flows and occupancies on each traffic link. The original state is given by statistical data describing the traffic demand between different sectors in the network.[13] Due to the state quantities' macroscopic character (i.e. they represent aggregated quantities) a model such as METACOR/METANET is usually described as macroscopic. Comparison of some standard output measures from these simulations with and without control clearly indicates that feedback control has a potential of improving the traffic situation vastly.[14]

In addition to the simulations, variable message signs managed by the control theoretical principles suggested by Papageorgiou have also been installed in the real world on freeway stretches in the outskirts of a few European cities. Evaluations of these tests also show that the control improves the over-all traffic conditions. In comparison with the improvements achieved in METANET/METACOR simulations however, these real-world improvements are rather modest.[15] Presumably, these less overwhelming results in the real-world testing are due to the fact that the METACOR/METANET-simulations, being very generalized and simplified representations of the real traffic systems, are comparably simple to control. One source of simplification in METACOR/METANET is that the traffic demand used often is a very rough average of the real traffic demand. Moreover, the topology of the simulated networks is typically very simple and the number of streets in the simulations rarely exceeds 100.[16] Finally, METACOR/METANET's differential equation framework makes the tuning of automatic feedback controllers rather straightforward, since a totally correct model of the system is available right away.

1.3 Microscopic multi-agent simulations

Macroscopic modeling is however not the only way of creating a synthetic representation of a traffic network. One other approach, called microscopic modeling,

---

[12] The VDS/VMS concept is for example described in Messmer (1994), p 691-692.
[13] METANET is presented in Messmer, A. & Papageorgiou, M., "METANET: a macroscopic simulation program for motorway networks" *Traffic Engineering and Control 31* (1990), pp 466-470. METACOR is defined in Elloumi, N., Haj-Salem, H., Papageorgiou, M., "METACOR: a macroscopic modelling tool for urban corridors" *Proceedings of TRISTAN II, vol. 1* (1994), p 135-150.
[14] Messmer (1994), p 699-701, Wang (2003), p 18-22.
[15] A real-world application from Glasgow, Scotland is discussed in Diakaki, C., Papageorgiou, M., McLean, T., "Applying integrating corridor control in Glasgow" *Proceedings of ASCE Fifth International Conference on Application of Advanced Technologies in Transportation Engineering* (1998), p 1-16. A test in Aalborg, Denmark is presented in Mammar, S., Messmer, A., Jensen, P., Papageorgiou, M., Haj-Salem, H., Jensen, L., "Automatic control of variable message signs in Aalborg" *Transportation Research Part C, vol. 4, No. 3* (1996), pp 131-150.
[16] Such generalizations are e.g. done in Messmer (1994) and Wang (2003).

takes the behavior of individual agents instead of aggregated data as a starting point. In microscopic traffic simulations, individual agents with individual plans travel through the traffic network and interact with each other according to some predefined, often rather simple, laws of motion. Advocates of microscopic simulations tend to maintain that such an individual agent approach is superior to macroscopic approaches in illustrating how urban traffic systems really work.[17] According to this point of view, no differential equation and no averaged macroscopic demand could ever describe the heterogeneity of large human populations and the complexity of human interaction in an urban traffic system without being too general or unforeseeably intricate. The micro-simulation approach, letting a huge number of agents interact in a comparably simple way, is thus the only possible way of generating tractable, yet realistic traffic simulations.[18]

Given the higher degree of complexity in microscopic multi-agent traffic simulations as a whole, it is extremely difficult (or even impossible) to transform the entire dynamics of such a simulation into a comprehensible differential equation framework. The lack of a straightforward mathematical model of the simulation makes it more difficult to implement feedback control in a micro-simulation framework than in a macroscopic simulation like METACOR/METANET. On the other hand, one could argue that the complexity of a micro-simulation makes it more similar to the real traffic system represented. Given this higher degree of resemblance, it is reasonable to assume that a successfully implementation of feedback control guidance in a microscopic simulation, is more likely to be successful also in reality, than control strategies that are developed and executed in a smoother macroscopic environment such as METACOR/METANET.

1.3.1 MATSim at Berlin's Technical University

Within the Institute for Transport Systems Planning and Transport Telematics at the Technical University in Berlin, a research group led by Professor Kai Nagel has been developing a microscopic multi-agent traffic simulation of Berlin since 2004.[19] The simulation tool, which is called MATSim, is written in JAVA and it has become more and more complex over the years. Very briefly, the simulation consists of a population of approximately 2 million agents (although only a sample of 10 % are actually being simulated) and a network with all *nodes* (intersections) and *links* (streets) in Berlin's road network. The total number of links and nodes is 30,000 and 11,500 respectively. Every agent has a *plan* specifying what *activities* it is going to carry out over the day as well as how the agent is going to travel between these activities. Such a trip between two activities is called a *leg*. The MATSim-plans are generated using a combination of

---

[17] It is predominantly urban traffic systems that might be better understood microscopically. For flow dynamics on highways, being less complex than urban traffic networks dynamics, macroscopic modeling is sufficient in most cases.

[18] Balmer, M., Cetin, N., Nagel, K., Raney, B., "Towards truly agent-based traffic and mobility simulations" in *Workshop on agents in traffic and transportation at Autonomous agents and multi-agent systems (AAMAS'04)* (2004)

[19] Before 2004, Professor Nagel did work at the MATSim-model at the Swiss Federal Institute of Technology (ETH) Zurich for five years. Before that he worked at the similar TRANSIMS model at Los Alamos Laboratories in the US since 1996.

statistical surveys and iterated best-path calculations.[20] At the moment only car traffic is modeled, implying that only agents traveling by car are taken into account. The plans of the agents provide information about what *routes*[21] all agents take as well as when they embark on their different trips. The movement of the agents through the network is thereafter determined by a set of rules regulating the traffic on each link as well as in every intersection. Finally, there is also a possibility for the agents to re-plan and take a new route the next day if they experience too long travel times on their original routes. This artificial intelligence feature ensures that the traffic gets more and more realistic over time.[22]

1.3.1.1 The Extended Cell Transmission Model

The methods for moving the agents through the simulated Berlin network realistically have been changed a few times over the years the MATSim-project has been in place. Presently, there are two possible ways of determining the network dynamics. At the one hand, there is a rather simple queuing model, where the traffic on each link is basically modeled like a first-in-first-out queuing system. The most positive aspect of this model is its simplicity, which makes it run smoothly with a minimum of calculations in each time step. Given the simplicity, it also models the traffic behavior surprisingly realistically.[23]

One of the most striking problems, using the queuing model is however that it is not able to model the build-up and dissolution of traffic congestion realistically. Using the queuing model, queues seem to dissolve from behind since all cars take a step forward instantly when a car is leaving the front of the queue. In reality, traffic congestion dissolve from the front, leaving the car in the very end of the queue standing still for the longest time.[24]

In order to avoid this and other problems associated with the queuing model, an alternative way to handle traffic dynamics amongst microscopic agents has been developed within the research group. This model's most striking feature is probably that the movements of the microscopic agents along the links are determined by a

---

[20] Generally speaking, the plans are generated in the following way. A very rough initial plans file is generated from statistical survey data. The simulation is run and the travel time on each link is registered. A fraction of the agents thereafter re-plan using best path calculations based on the registered link travel times. The simulation is then run once more and new travel times are generated, which are once again used for re-planning etc. After sufficiently much iteration, the plans there will not be necessary for the agents to re-plan any longer. The plans file obtained at that stage ensures that the entire traffic system is in a rough Nash Equilibrium. Nagel (2000).

[21] The concepts "leg" and "route" are very similar and easy to confound. Basically, every leg has a route but also a few other parameters such as departure time and mode of travel. The mode of travel has so far always been car.

[22] Balmer (2004).

[23] Simon, P. M., Esser, J., Nagel, K., "Simple queuing model applied to the city of Portland" (1999) published on the Institute for Transport System Planning and Transport Telematics, TU Berlin website: http://www.vsp.tu-berlin.de/archive/sim-archive/publications (2007-01-08).

[24] Kerner, B. & Rehborn, H., "Experimental properties of complexity in traffic flow" *Phy. Rev. E*, *5*, (1996).

macroscopic flow model. The macroscopic model, which is an extended version of Carlos F. Daganzo's Cell Transmission Model[25], does however only determine the behavior on each link. At the intersections, it is instead every agent's individual plan that determines which link it shall proceed to. In order to calculate the dynamics on the links correctly, it is also necessary for the macroscopic model to know how much flow that is coming into each link; that is, the flow must be split appropriately in the intersections. In the ECTM-model that is done by counting how many of the microscopic agents that travel left and right respectively at each crossing.[26] Figure 1 shows how the interaction between the microscopic and macroscopic level structurally works.

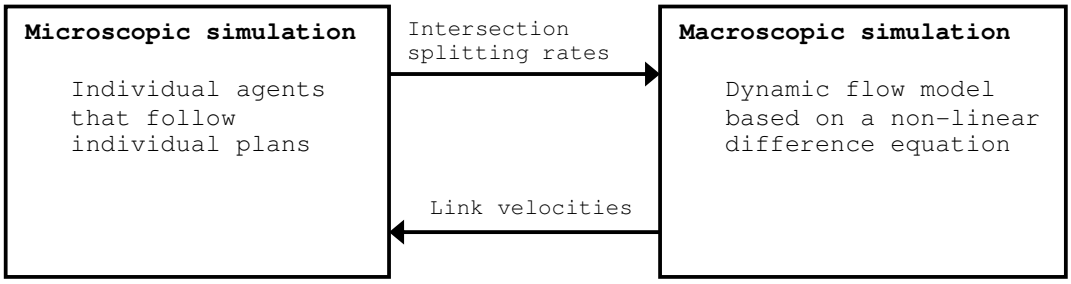

Figure 1. Interaction between the microscopic and macroscopic parts of the ECTM-model.

The ECTM-model can thus be characterized as a mixture of a microscopic and a macroscopic model. In this way, it hopefully combines the best aspects of these two simulation-approaches. On the one hand, there are microscopic agents that travel through the simulation according to individual plans. This makes it possible to model the great heterogeneity in a population like that of Berlin. On the other hand, the macroscopically modeled link dynamics make the modeling of for example congestion pattern more realistic than the one achieved by simpler models like the queuing model.[27] Moreover, the ECTM-model also has the advantage of being differentiable, facilitating advanced analysis and control.[28] On the other hand, the ECTM-model, being more complex, is slower than the Queuing-model. The decrease in speed is rather modest though, not exceeding a factor of ten.

---

[25] The Cell Transmission Model (CTM) is presented in Daganzo, C. F., "The Cell transmission model: A dynamic representation of highway traffic consistent with hydrodynamic theory" *Transportation Research Part B, vol. 28B, No. 4* (1994), pp 269-278 and developed in Daganzo, C. F., "The Cell transmission model, part II: Network traffic" *Transportation Research Part B, vol. 29B, No. 2* (1995), pp 79-93.

[26] The ECTM-model is presented in Flötteröd, G. & Nagel, K., "Some Practical Extensions to the Cell Transmission Model" Conference Paper from IEEE ITSC, Vienna, 2005. The method of counting microscopic turning moves was added in Flötteröd, G. & Nagel, K., "Simulation and optimization of trajectories in a congested network" (2006) published on the Institute for Transport System Planning and Transport Telematics, TU Berlin website: http://www.vsp.tu-berlin.de/archive/sim-archive/publications (2007-01-26).

[27] Daganzo (1994), 273-276.

[28] Flötteröd (2005)

Within this master thesis project, the ECTM-model has been used consistently in the simulations. The reason for that was simply that the ECTM was the model made available at the department. Nevertheless, considering that this model gives slightly more realistic traffic behavior than the Queuing model, it would probably have been the preferred model also if there would have been an option available.

1.3.1.2 The COOPERS-project

Prior to the beginning of fall 2006, guidance of any kind had not been part of the MATSim-project. Intentions to implement some kind of guidance as well as other telematics measures had however been present for quite some time. In one sub-project involving parts of the research group, the implementation and analysis of an in-car information provision device, called a COOPERS-device, has been planned. The development of this device is part of the European Union COOPERS-project, standing for "Co-operative Systems for Intelligent Road Safety". The general aim of this project is the "[e]nhancement of road safety by direct and up to date traffic information communication between infrastructure and motorized vehicles on a motorway section".[29] The project is divided into several sub-projects, one of them taking place at the Institute for Transport Systems Planning and Transport Telematics at the Technical University in Berlin. Furthermore, the information provision between the device and the traffic management is supposed to be bi-directional. That means that information can be provided both from the infrastructure to the vehicles and from the vehicles to the infrastructure.[30]

Assuming that every agent in the simulation had such a device installed in its simulated car would make it possible to analyze if and how the traffic situation in Berlin could be improved, if one provided a certain fraction of the agents with guidance via the devices. The information given to this kind of device is broadcasted from a traffic management central. A GPS-navigation system in the device keeps track of the position of the cars, making it possible to filter the information such that the cars only get the information relevant in their present location. One could also assume that the drivers give the device information about the trip they are about to a make upon leaving. Using this personalized information for filtering the broadcasted information, the information provision to the drivers could get even more specific. Guidance information could then be given only to those drivers that are bound for a relevant destination.

1.4 The aim of the thesis

The overall and general aim of this master thesis project is to investigate if automatic feedback control has a potential of improving the traffic situation in the Berlin metropolitan area. More specifically, this is done by means of controlling the traffic in the MATSim representation of Berlin via in-car COOPERS-devices that all agents are assumed to have in their cars. Given MATSim's microscopic character and the high complexity of the network and population used in the Berlin simulation, it is reasonable

[29] The COOPERS Website: http://www.coopers-ip.eu (2007-01-08).
[30] ibid.

to assume it to be more difficult to implement successful automatic feedback control guidance in this framework than in a macroscopic framework such as METACOR/METANET. On the other hand, if such guidance could be implemented successfully, the control would presumably work better than the METACOR/METANET-guidance if it is applied in the real world.

This overall aim is split up into two separate sub-aims. The first of these is to write software that makes it possible to control the traffic in the MATSim-framework by means of COOPERS-devices directed by different automatic feedback control strategies. This software must be flexible enough to facilitate different control strategies. Moreover, it is of importance that the simulated traffic behaves realistically when the agents receive the COOPERS-guidance.

The second sub-aim of the thesis is to use the guidance for controlling the simulated traffic in Berlin. The aim of the control is to achieve and keep a Nash Equilibrium situation on two alternative routes connecting one origin with one destination. A Nash Equilibrium is a game theoretical concept defined as a situation where no agent has anything to gain from choosing a different option unilaterally.[31] In a traffic situation with two alternative roads, this definition corresponds to a situation where all agents are traveling on the least costly, i.e. the fastest,[32] route. A sufficient condition of a Nash Equilibrium is therefore that the travel times on both roads are equal. In that case, both roads can be said to be fastest and all travelers are therefore obviously using a road that is the fastest possible.

Equal travel times is however not a necessity. A Nash Equilibrium is also achieved when one of the roads are faster, provided that the entire traffic uses this route. This is the usual situation in a normal day scenario if the control is applied to one main road and one slower, alternative road that is not normally used. In such a situation the travel times on the two routes are not equal since it is always faster to use the main road as long as nothing extraordinary happens. Nevertheless, as long as all travelers use the faster main route, the system is still in a perfect Nash Equilibrium.[33]

1.5 Disposition

The method for fulfilling the aims stated above is discussed in Chapter 2. In a first subchapter, the different functionalities that had to be implemented are briefly described. Control theoretical aspects of the traffic system, such as the different signals studied, the control strategies applied and the methods for actuation of the control signal is described in Chapter 2.2. For readers already familiar with basic control strategies, parts of this chapter might seem elementary and Section 2.2.2 to Section 2.2.4 could be left out. In Chapter 2.3, the simulation settings of the two networks used

[31] "Nash equilibrium", Wikipedia (2007).
[32] I assume throughout this thesis that the cost of traveling a certain road stretch is defined as the travel time. This is a common assumption in traffic engineering.
[33] The Nash equilibrium concept is for example used in this way in Messmer (1994), p 696.

for the simulations are described. This chapter also includes a section about the evaluation measures that were used to compare different control strategies.

The Java-implementation is discussed in Chapter 3. The classes responsible for measuring, feedback and guidance and perceiving and reacting are discussed in three different subchapters. The simulation results are covered in Chapter 4. Simulations on a small test network are presented in Chapter 4.1 whereas simulations on the larger Berlin network are discussed in Chapter 4.2. Both these chapters have separate sections covering a “Normal case”, an “Accident case” and robustness tests. In Section 4.1.4 the conclusions drawn from the smaller network result in a set of guidelines for control design in traffic networks. In Section 4.2.4 the experiences from the simulations on the larger Berlin network, make it possible to re-evaluate some of these guidelines.

In Chapter 5, some main conclusions are drawn. Furthermore, an outline for how this sort of guidance could be implemented in the real world and some suggestions for future research are proposed in this concluding chapter.

2 Method

2.1 Necessary additional functionality

In order to fulfill the aims stated above, some extra functionality, not already available in MATSim had to be added. In an initial phase of the project, a set of JAVA-classes carrying out these tasks thus had to be implemented. One necessary function was to measure the control output. As will be discussed in Chapter 2.2, the travel time deviation between two pre-defined roads is a relevant output quantity to keep track of when a Nash Equilibrium on that road couple is desired. Hence, implementation of software being able to measure this output was necessary.

A feedback functionality connecting the output measures of the traffic conditions with the input guidance broadcasted to the COOPERS-devices was also necessary. This particular part of the JAVA-code needed to be flexible in terms of control strategies so that suitable control strategies could be adopted in different control situations. Furthermore, this software also had to obey to constrains given by the COOPERS-technology as well as fundamental aspects of human psychology. Examples of such constrains could be that the COOPERS-technology does not allow for sending several messages at the same time and that people would not find it meaningful with a guidance messages that change too frequently.

Finally, another obvious functionality that had to be implemented was a software representation of the COOPERS-devices and the way the agents perceive and react to the information provided by these. Similarly to the real-world device, this implementation had to be able to filter the guidance broadcasted depending on the current position of the car as well as the route the agent is about to travel.

A full description of how this functionality was actually implemented is found in Chapter 3.

2.2 Control strategies

The literature covering automatic feedback control is extensive and the amount of strategies one could potentially use is large. However, a majority of the strategies requires a linear differential equation model of the system. Given such a model, a huge framework for sophisticated analysis and tuning of the control loop is available, facilitating very exact behavior of the control output. Since no such model is available for the multi-agent simulation controlled within this master thesis project, a vast part of the control theoretical research field becomes obsolete. In fact, only those strategies that could be applied “blindly”, i.e. without an explicit model of the system, could be taken into consideration. Typically, blind feedback control can only be carried out using very basic and simple control strategies. Due to this constrain, the control

strategies applied to the simulated traffic was limited to bang-bang control, proportional control and proportional and integral control.[34]

2.2.1 Signals and notation

The simulations in MATSim are all discrete in time, having a time step of one second. In the remaining section, this discrete time variable will be denoted *t*. It is important to note that this discrete time step implies that the *system* that is to be controlled is time discrete. That is, although MATSim represents a real traffic system with continuous time, the system controlled in this project is not this real world but *the simulation itself* which is a truly time discrete system.

As already mentioned, the control aims at achieving and keeping a Nash equilibrium on two alterative routes connecting a common origin with a common destination. A sufficient condition for a Nash equilibrium is that the travel time on both routes are equal and the difference between these two travel times has therefore been seen as the relevant output of the system. Remembering that equal travel times is not a necessary condition for Nash equilibrium (Chapter 1.4), it might seem strange that the output does not also take the number of vehicles traveling on each route into consideration. The reason for choosing the travel time deviation as output is partly due to the fact that this seems to be the standard approach used by other researchers trying to achieve and keep Nash equilibriums.[35] In addition, control strategies based on this output does not run the risk of distorting Nash equilibriums with one faster route that everyone is using. One might fear that the control strategy would make that faster route slower, in order to decrease the travel time difference between the two routes. In practice, it may be true that the control strategy *tries* to make the faster route slower However, the only way of actually making the fast route slower is to guide more traffic into that route. That is obviously practically impossible, since the entire traffic is already using the faster route.

In the remaining sections of this report, the output from the system, i.e. the travel time deviation on the two routes, will be denoted $\Delta\tau_R(t) = TT_R^1(t) - TT_R^2(t)$. When more general control theoretical aspects are discussed, the same signal is sometimes denoted $y(t)$. The route travel time $TT_R^j(t)$ at time *t* on a route *j* is for practical reasons defined as:

$$TT_R^j(t) = \sum_{i \in J} tt_R^i(t) \qquad (1)$$

J in this formula represents the set of links constituting route *j*. The quantity $tt_R^i(t)$ is the link travel time experienced by the vehicle, which most recently exited link *i*. The reason for using this formula is that link travel times, in contrast to route travel times

---

[34] Theoretically, it would also have been possible to use proportional, integral and derivative control (PID-control). In practice, it is usually really hard to tune such a controller, containing three control parameters without an explicit model. It is also questionable how much one could gain from using derivative control, given the complexity and noisiness of the traffic system being controlled.

[35] This output is for example used in Messmer (1994), p 696-699.

are readily available from the multi-agent simulation. The subscripted *R* stands for "reactive" following the standard notation in traffic engineering research articles.[36] The reference value, which the output should be controlled towards, is always $r(t) \equiv 0$.

The input to the system is defined as the way the traffic management central tries to split the traffic flow at the intersection at the beginning of the two alternative routes.[37] This signal is either denoted $\beta(t)$[38] or $u(t)$ when the discussion is not traffic specific. $\beta(\mathrm{t}) = 1$ implies that all traffic is to be directed into route 1 whereas $\beta(t) = 0$ means that all traffic is guided into route 2. Hence, $\beta(t) = 0.7$ implies that one desires to guide 70 % of the traffic into route 1 and 30 % into route 2. Obviously $\beta(t) = u(t) \in [0,1]$ always holds for the input.[39] I the rest of the text, it is assumed that $\mathrm{u}_0 \in [0,1]$ is the splitting of the traffic when no guidance is applied.

2.2.2 Bang-bang control

An extremely simple and intuitive way to control the system is by always guiding all the traffic into the presently faster of the two roads. Mathematically that means that the controller sets the input to the system according to the following control algorithm.

$$\begin{aligned} &If: y(t) < 0, then: u(t) = 1 \\ &If: y(t) > 0, then: u(t) = 0 \\ &If: y(t) = 0, then: u(t) = u_0 \end{aligned} \qquad (2)$$

A control strategy based on this "all-or-nothing algorithm" is denoted bang-bang control (or relay-control). They main virtue of this strategy is its simplicity. In contrast to more complex control strategies, it does not include any parameters that have to be tuned differently in different scenarios. This system-independent aspect of the control strategy also makes it comparably robust if the system behavior is changing. In contrast to other control strategies, a bang-bang controller always uses the same algorithm and is therefore never badly tuned. Furthermore, except for the rather rare case when $y(t) = 0$, Bang-Bang control avoids the problem of how to materialize a splitting rate that is neither $\beta(t) = 0$ nor $\beta(\mathrm{t}) = 1$.[40]

There are however also a few drawbacks associated with bang-bang control. The main problem has to do with the fact that bang-bang controllers tend to control too strongly, resulting in oscillating system behavior. The reason for this is that the controller sends *all* agents into the fastest road also when the travel time difference is very close to zero. Typically, the road into which all agents are directed soon gets jammed, leading to

---

[36] In Section 2.2.5, another travel time measurement called "predictive" is introduced. The difference between reactive and predictive travel times is discussed in Wang (2003), p 3-4.
[37] Exactly how this desirable splitting is to be achieved by the intrinsically binary guidance information provided by the COOPERS-devices is a non-trivial question discussed extensively in chapter 2.2.6.
[38] This notation seems to be a standard notation for control splitting rates. It is for example used in Messmer (1994), Wang (2003) and Kotsialos (2002).
[39] As will be shown in several sections later on, these boundaries on the input does constrain the control performance in many ways.
[40] Messmer (1994), p 697-698.

increasing travel times after a time lag. Following this travel time increase, the Bang-Bang controller starts guiding all the traffic into the other road instead, until that road gets jammed and slow. In a system controlled by a Bang-Bang controller, it is therefore common that the system oscillates, also in situations where the output would have been close to zero without any control at all. In other words, the stability of the system can not be guaranteed.[41]

2.2.3 Proportional control (P-control)

The oscillating behavior normally observed in a system controlled by bang-bang control is caused by the fact that the controller does not at all consider the magnitude of the output error when calculating the input. That is, a very small positive value of $y(t)$ is considered equally as considerably larger $y(t)$-values. One way of getting around this problem is to apply a control with an input that is proportional to the control error. Considering that $u(t) \in [0,1]$ must hold, a proportional controller obeys the following algorithm.

$$\begin{aligned} &\hat{u}(t) = -K_P y(t) \\ &If : \hat{u}(t) \geq 1, then : u(t) = 1 \\ &If : \hat{u}(t) \leq -1, then : u(t) = 0 \\ &Else : u(t) = u_0 + \psi[\hat{u}(t)] \cdot [1 - u_0] - \psi[-\hat{u}(t)] \cdot u_0 \end{aligned} \qquad (3)$$

In this algorithm, $\psi(\cdot) = \max(0, \cdot)$ ensure that $u(t) \in [0,1]$. The parameter K_P in the controller is a parameter that one chooses freely. If K_P is chosen large, the control gets faster but the risk of oscillating system behavior increases. If K_P is really large in comparison with the magnitude of $y(t)$, the proportional controller works as a bang-bang controller in practice. In contrast, small K_P-values make the system reaction slower but non-oscillating. In order to achieve good system behavior, it is thus of great importance to tune the K_P-value depending on the system's characteristics.[42]

An appropriately tuned proportional controller is generally superior to a bang-bang controller, especially when it comes to oscillations and stability. It is also easy to implement and mathematically very simple. Moreover, its one and only control parameter is often possible to tune without an explicit model of the system by means of some kind of trial-and-error method. This quality makes it a suitable strategy for the system controlled in this thesis. The main drawback to this type of control is however that it can not guarantee that the control output tends towards its reference value asymptotically. In fact, systems controlled by proportional controllers very often have static errors, meaning that the output is kept constant at a non-desirable level.[43]

The following example illustrates how this could happen:

---

[41] ibid.
[42] Glad & Ljung (1989), p 13-14, 41-44.
[43] Glad & Ljung (1989), p 14.

Assuming that $K_P = 0.1$, $y(t) = 5$ and $u_0 = 0.5$ it follows that $\hat{u}(t) = -0.5$ and $u(t) = 0.25$. That is indeed a reasonable choice of the input; the output is too big and one should therefore use an input that is closer to zero than one. It is however possible that the system dynamics implies that $u(t) \equiv 0.25$ generates $y(t) \equiv 5 \neq 0$ asymptotically.[44] In that case, the closed system has reached a stable fix point situation that is not the desired one; in other words, the system has a static error.

2.2.4 Proportional and integral control (PI-control)

The standard solution to the static error problem discussed above is to add an integrating part to the proportional controller, making it a proportional and integral controller (PI). For continuous time systems, this means that also the integral of the prior output is taken into account when the intermediate value $\hat{u}(t)$ is calculated. For discrete systems, like the one controlled in this project, the integral is replaced by a sum of prior output values, giving a control algorithm that looks like this:

$$\begin{aligned} &\hat{u}(t) = -K_P y(t) - K_i \sum_{i=0}^{t} y(i) \\ &If: \hat{u}(t) \geq 1, then: u(t) = 1 \\ &If: \hat{u}(t) \leq -1, then: u(t) = 0 \\ &Else: u(t) = u_0 + \psi[\hat{u}(t)] \cdot [1 - u_0] - \psi[-\hat{u}(t)] \cdot u_0 \end{aligned} \qquad (4)$$

The addition of the integrating part usually eliminates the static error problems. The reason for that is that there can be no fix point at $y(t) \equiv k, k \neq 0$ since in such a situation the sum would grow bigger and bigger, continuously changing the value of the input.[45] The only risk of getting a static error situation with such an integrating part in the controller is if inputs with sufficiently large magnitude can not be generated since the input is bounded in one way or another. In this project $u(t) \in [0,1]$ always holds. Static errors can thus remain in the system controlled in this project also when PI-control is applied, if $u(t) \equiv 0 \Rightarrow y(t) \equiv k, k > 0$ or $u(t) \equiv 1 \Rightarrow y(t) \equiv k, k < 0$. That is, the control error does not disappear even though the controller controls as hard as it possibly can.[46]

As is easily seen, a P-controller is a special case of a PI-controller with K_i set to zero. Compared with a P-controller, this extra control parameter gives a PI-controller one additional degree of freedom. Hence it follows that a PI-controller has a potential of being tuned more precisely than a P-controller. At the same time, tuning of two parameters simultaneously also requires more work. Especially when the tuning is done without an explicit model by means of trying a set of reasonable parameter

---

[44] For the traffic system controlled in this master thesis, this could be the case for example if route 1 is longer than route 2.
[45] Glad & Ljung, p 14, p 44-45.
[46] Glad & Ljung, p 227-228.

combinations, the complexity of the tuning algorithm is increased from n to n^2 simulation runs.[47]

The general PI-algorithm stated above, is however rarely used in real-world applications. Instead, the algorithm is often being executed using differentiated data. The differentiated PI-control algorithm typically looks like this:

$$\begin{aligned}
&\Delta u(t) = -K_P(y(t) - y(t-1)) - \frac{1}{T_i}y(t) \\
&If: \Delta u(t) + u(t-1) \geq 1, then: u(t) = 1 \\
&If: \Delta u(t) + u(t-1) \leq -1, then: u(t) = 0 \\
&Else: u(t) = u_0 + \psi[\Delta u(t) + u(t-1)] \cdot [1 - u_0] - \psi[-(\Delta u(t) + u(t-1))] \cdot u_0
\end{aligned} \qquad (5)$$

This second algorithm (5) is basically nothing but a differentiated version of the first algorithm (4) with slightly changed behavior at the boundaries and a new control parameter $T_i = 1/K_i$.

The use of algorithm (5) does avoid a problem called *integrator windup* that otherwise often follows when an integrated part is introduced in the controller and the input signal is bounded. This problem is commonly observed in situations like the one described above, where also controllers with integral parts are unable to eliminate static errors. As already mentioned, this could only happen if the input is stuck to a boundary. When a static error is observed over a long period of time, the magnitudes of the sum in (4) as well as $\hat{u}(t)$ tend to grow very large. In a situation where $y(t)$ later, due to changing conditions in the system, changes sign, the output will be allowed to be at a non-zero level for a long time before the sum in (4) returns to zero. $\hat{u}(t)$ will therefore be kept large and $u(t)$ will stick to its boundary-value undesirably long. The result of this phenomenon is typically strong and long-lasting overshoots in the output. [48]

In contrast to a PI-controller that uses control algorithm (4), windup is not a problem for PI-controllers implemented with (5). The reason for that is not the differentiation of the data[49] but rather the way the input is handled at the boundaries. In contrast to (4) no quantity is updated in (5) if $u(t)$ is at a boundary. That means that the controller only takes prior outputs into account *if these outputs were generated when the input was not at a boundary*. In a situation where there has been a static error for a period of time and the output suddenly changes sign, the input will thus leave its boundary-value much

[47] In practice, the increased complexity arises when in the tuning process, one loop, is replaced by two nested loops.
[48] Glad & Ljung, p 228-229.
[49] A straightforward differentiation can not change anything, since the equation will obviously be equivalent.

faster.[50] The problems with overshoots will therefore be less severe if algorithm (5) is used instead of (4).

As will be shown in Chapter 4.2, static error situations with the input at a boundary are common for the system controlled in this project. This is for example the case in the Nash equilibrium situation when one road is faster and all travelers use that road. In order to avoid the problems associated with integrator windup, the PI-controllers used in this project are consistently implementations of control algorithm (5) instead of (4).

2.2.5 Predictive measurements

Dynamical systems containing time lags are often problematic to control using automatic feedback control strategies. A time lag refers to a situation where the input at one time instance does not start influencing the output immediately. That is $u(t_1)$ has no effect on $y(t)$ when $t < t_2 = t_1 + T$, T being the time lag. In a time lag situation, a feedback controller basing its input to the system on the current output is likely to be too late in its control decisions, leading to oscillating system behavior. Typically the controller, seeing no results of its adjustments, adjusts too strongly making re-adjustments later on necessary. The situation obviously gets more severe, the longer the time lags are.[51]

In traffic networks, time lags are ubiquitous. In system with route guidance given at a certain intersection as the input and an output determined by travel times, the sizes of the lags depend on the lengths of the routes. Obviously, when the routes are long, the time before the guidance has an impact on the travel times is also long. These time lags could potentially turn out being very problematic for the controllers used in this thesis, making it relevant to consider some ways of coping with time lag problems.

The standard method used when a system with time lags is to be controlled is to use predicted values instead of current values as the control output. That means that $y(t_2)$, which is the first output that is influenced by $u(t_1)$, is predicted by means of running a model of the system with the most current input held constant in $t_1 < t < t_2$. $y(t_2)$ is then used in the control algorithm in one way or the other.[52]

Unfortunately, no model for prediction was realizable within the available project time for the multi-agent simulation controlled within this project.[53] The predictive control

---

[50] That does not necessarily mean that the boundary will be left immediately when the sign of the output changes. The controller is still integrating and the sum of outputs that were generated before the input reached the boundary does obviously still matter.

[51] Technically speaking, the time lag decreases the phase margin, pushing the system closer to instability. Time lags are discussed in Glad & Ljung (1989), p 105-107.

[52] A common control structure that bases its control on a prediction of the output is the Otto Smith controller discussed in Glad & Ljung (1989), p 134-136. A more general approach is called "Internal Model Control" (IMC), which is discussed in Glad & Ljung (2003), p 256-261. A variant of this control structure is applied to a traffic system in Wang (2003).

[53] In MATSim there is a certain mode that generates accurate predictions of future system performances. Unfortunately, the additional implementation necessary for using these predictions in the control turned out being too complicated and the predictive mode was thus never used.

approaches discussed above were thus not possible to use. Seeing that the simulation facilitates two different ways of calculating travel times on the two routes, there are however ways to make the control at least semi-predictive. As already mentioned in Section 2.2.1, the standard way of calculating the system output uses the reactive route travel time calculation defined in (1).

The other method for route travel time calculation is defined by the following formula:

$$TT_P^j(t) = \sum_{i \in J} tt_P^i(t) \qquad (6)$$

j is also here the number of the route and *J* is the set of all links making up route *j*. $tt_P^i(t)$ is here denoting the predictive travel time on link *i*. The predictive link travel time is calculated as

$$\begin{aligned} &\overline{tt}_P^i(t) = \frac{\eta^i(t)}{f^i(t)} \\ &If: \overline{tt}_P^i(t) > tt_{ff}^i, then: tt_P^i(t) = \overline{tt}_P^i \\ &Else: tt_P^i(t) = tt_{ff}^i \end{aligned} \qquad (7)$$

where $\eta^i(t)$ is the number of cars on link *i* at time *t* and $f^i(t)$ is the maximum outgoing flow allowed by the traffic situation on the downstream link (cars/second) at time *t*. tt_{ff}^i is the travel time on link *i* at free flow which in (7) is a lower bond for $tt_P^i(t)$. Assuming that the flow on a link is constant, this way of measuring the travel times gives a good prediction of the travel time of the car just entering the link.[54] The use of TT_P instead of TT_R as the control output thus has a potential of making the controller more similar to a predictive controller. Hence, time lags might be handled in a better way. Furthermore, TT_R and TT_P are both measurements that are possible to obtain in a real world traffic network where the cars are equipped with COOPERS-devices. Given the ability of the device to keep track of its position, reactive travel times are easily evaluated by measuring the time it takes for each car to travel from the beginning to the end of each link. Predictive travel times are also easily available since occupancy as well as outgoing flow is given by straightforward counts of the number of cars entering and leaving the link.[55]

It is however important to notice that an output generated using the predictive route travel times TT_P is not identical to the output normally used by a predictive controller. That is

$$\Delta\tau_P(t_1) \neq \Delta\tau_R(t_2); t_2 = t_1 + T \qquad (8)$$

---

[54] This is intuitively logical. If the flow is constantly f all η cars have left the link after η/f seconds.

[55] Flow and occupancy can also be measured without the COOPERS-devices, using inductive loops at certain points in the network. In contrast to reactive travel times, the measurement of predictive travel times is thus not dependent on the COOPERS-technology.

The main reason for that is that the definition of a certain time lag *T* is somewhat ambiguous, given the system studied and the way of calculating the system output. It is not at all easy to say when $TT_R(t)$ will be affected by a guidance at t_1. In fact, it seems reasonable to assume that $tt_P(t_1)$ is a good prediction of $tt_R(t_2)$ for the very first link on the route, since $tt_P(t_1)$ is actually predicting the reactive travel time of the car that is just about to enter the link at time t_1, which in case of the first link is the guided traffic. For all other links, it is harder to interpret what the predictive travel times actually measure. The predictive route travel time, being the sum of all predictive link travel times, should therefore not be used unreservedly.

Given these problematic properties of the predictive travel times, one should be cautious using $\Delta\tau_P(t)$ instead of the normal $\Delta\tau_R(t)$ in the control loop. $\Delta\tau_P(t)$ does indeed include a predictive aspect that might be useful for coping with time lags but since the prediction is not exactly the one one would wish to have, no improvements can be guaranteed. Hence, instead of seeing the predictive control as something obviously preferable, the use of predictive instead of reactive travel times has been considered as an extra degree of freedom that has been used in a trial-and-error fashion when tuning the controllers.

2.2.6 Continuous input - binary guidance

The control signal $\beta(t)$ generated by the controller is a number between zero and one but the guidance given to the cars can only be either "take route1" ($\beta = 1$) or "take route 2" ($\beta = 0$) at every given time instance.[56] The problem of how to generate a continuous input from such intrinsically binary guidance information is a non-trivial problem that could be solved in several ways. The way of solving this problem chosen in this project is based on the idea that an aggregated sequence of binary variables resembles a continuous variable. $\beta(t)$ is thus actuated as a sequence of guiding messages that is used over a certain time horizon.[57] The number of messages in the sequence is a highly significant variable. This parameter, called *c*, can be chosen freely and can thus be seen as a control parameter that needs to be tuned.[58] Furthermore, the time period over which each message is broadcasted is also important. In this project, this time slot, which is denoted T_{fix}, has been assumed to be 30 seconds. The reason for this choice is that if the guidance is changed more frequently, the agents would presumably find the guidance annoying and stop paying attention to it. The time horizon over which the sequence of messages is kept constant is obviously $T_{hold} = T_{fix} \cdot c$. After that time interval, a new sequence of guidance messages is constructed according to the value of $\beta(t)$ at that time.

An example with $\beta(t) = 0.4$, $c = 3$ and $T_{fix} = 30$ shows how this part of the control works: [59]

---

[56] This is due to the COOPERS-infrastructure's inability to transmit more than one message at a time.
[57] The same way of generating the input is for example used in Messmer (1994), p 695.
[58] Exactly how this parameter was tuned is discussed in Section 4.1.1.4.1.
[59] As described in chapter 4.1.1.4.1, $c = 3$ turned out to be a good choice of this parameter..

$\beta(t) = 0.4$ implies that the traffic is to be split so that 40 % are traveling on route 1 and 60 % on route 2. This splitting rate is to be achieved using three different messages, each shown for 30 seconds. The order of the messages is not considered to be of interest, since only the aggregated result over the entire time horizon matters. Hence, the alternative ways of guiding the traffic over the upcoming 90 seconds are {1,1,1}, {1,0,0}, {1,1,0}, {0,0,0}. The mapping of $\beta(t) \rightarrow \{\beta_1(t), \beta_2(t), \beta_3(t)\}$ logically looks like this: [60]

$$\begin{aligned} &\beta(t) \in [0,0.25) \rightarrow \{0,0,0\} \\ &\beta(t) \in [0.25,0.5) \rightarrow \{1,0,0\} \\ &\beta(t) \in [0.5,0.75) \rightarrow \{1,1,0\} \\ &\beta(t) \in [0.75,1] \rightarrow \{1,1,1\} \end{aligned} \qquad (9)$$

Given this mapping, $\beta(t) = 0.4$ would generate a sequence of guidance messages over the upcoming 90 seconds that guides the traffic towards route 1 during 30 seconds and towards route 2 during 60 seconds.

The conversion of the continuous input $\beta(t)$ into a sequence of binary guidance messages obviously worsens the controller's ability to control the system output desirably. A variable $\beta_{real}(t)$, representing the actual splitting of the traffic over the current T_{hold}-interval is needed if this limitation is to be understood properly. $\beta_{real}(t)$ is defined like this:

$$\beta_{real}(t) = \frac{\sum_{i=1}^{c} \beta_i(t)}{c} \qquad (10)$$

One of the most problematic aspects associated with the actuation procedure is that although the control input $\beta(t)$ is updated as often as the system time step (one second), the sequence of guiding messages (and $\beta_{real}(t)$) is updated much more rarely. As a matter of fact, the variable $T_{fix} = 30s$ constitutes a lower bound for the time interval T_{hold} between these updates. Keeping the input to the system constant over such a long period of time obviously makes the controller slower and less exact. Technically speaking, the lower sampling of the input decreases the phase margin of the loop gain, leading to a closed loop system with smaller stability margin.[61]

From the line of arguments above, it is tempting to conclude that the variable *c* should always be set to 1. In this way, one would minimize T_{hold} during which $\beta_{real}(t)$ is kept constant as well as the stability problems associated with large such intervals. Unfortunately, choosing $c = 1$, leads to a situation where $\beta_{real} \in \{0,1\}$. This obviously

---

[60] $\beta_i(t) \in \{0,1\}$ is here the guidance at position *i* in the message sequence defined most recently at time *t*. Since the order of the messages does not matter, the positions of the messages do not necessarily correspond to the temporal order in which the messages are actually broadcasted.

[61] Glad & Ljung (1989), p 221.

implies very rough approximations of $\beta(t)$ and in practice, such a setting would make any controller behave like a bang-bang controller.

In order to get the preciseness necessary for achieving the advantages of P- or PI-control described in chapter 2.2.3-2.2.4, the value of *c* thus has to be larger than one. As an example, $c = 5$ extends the possible real inputs to the system to $\beta_{real} \in \{0,0.2,0.4,0.6,0.8,1\}$. This obviously makes the approximations of $\beta(t)$ less harsh, but at the same time, the time interval T_{hold} during which $\beta_{real}(t)$ is kept constant is extended from 30 seconds to 150 seconds. Hence, the choice of *c* will always be a trade-off between accuracy and fastness and it is never trivial to tune this parameter optimally.

2.3 Simulation settings

Running the MATSim micro-simulation using the full Berlin network with approximately 30,000 links and a population of some 200,000 agents is a computationally costly process that makes the tuning of control parameters by means of testing all relevant combination impossibly time consuming. The simulations carried out within this master thesis project have therefore been run on a reduced version of the Berlin network including only the roughly 2500 links having a capacity of more than 2000 cars per hour[62] and a population of approximately 170,000 agents[63]. Also this reduced Berlin network was however way too clumsy to use in the process of software implementation as well as when the basic functionality of the controller was to be tested. As an example, loading the population alone takes about 5 minutes on a 1.8 GHz, 2048 MB computer. Furthermore, the simulation speed-up on the same machine is approximately 50, which means that 50 simulation seconds can be simulated during one real world second. Each of the 6 hour simulation runs, used for the final evaluation, thus take roughly 12 minutes in total. In an initial phase of the project, a much smaller test network was therefore used to speed up the process.

2.3.1 Simulations on the small test network

The small network used for initial testing was designed to make it easy to check that the new software worked properly. Moreover, the traffic situation in the network was deliberately shaped so that interesting traffic scenarios could be generated with ease. Figure 2 shows a screenshot of the network with the link identity numbers written out.

[62] Some smaller links later had to be added in order fill the gaps between these high-capacity links.
[63] This is a sample of 40% of the population that is relevant on this reduced version of the Berlin network.

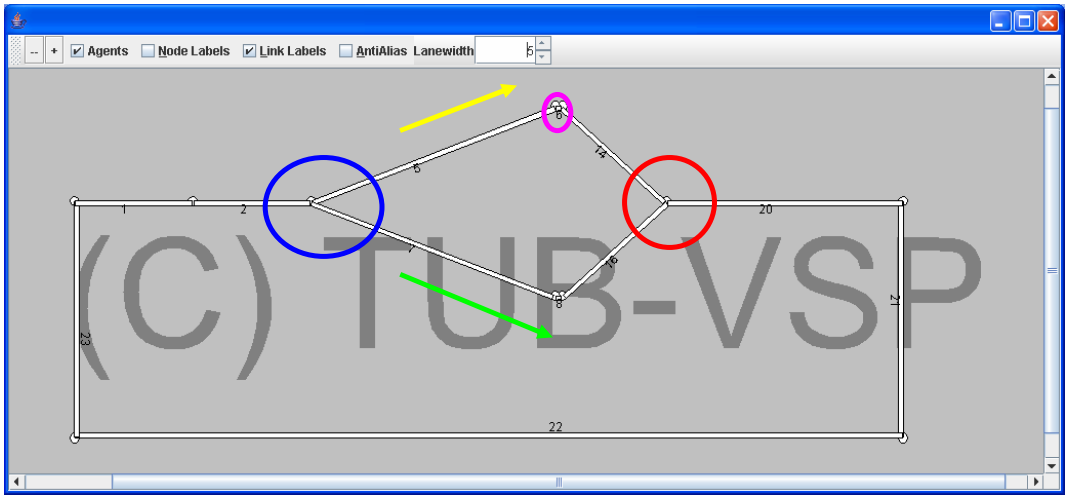

Figure 2. The basic structure of the small test network. Guidance is given in the blue circle to cars traveling towards the red circle. Route 1 is yellow, route 2 is green. An accident can be generated in the violet circle.

The 6000 agents in the population used for simulations on this network, all start from link 1 traveling to link 21. The first agent leaves link 1 at 07:00:00 in the morning. The time between each subsequent departure after agent 1 is exponentially distributed with an expected value of one second, all departure times being rounded to the nearest whole second. The expected departure time of the last agent is therefore 08:40:00 in the morning. Moreover the route each agents takes is randomized so that 50% of the agents use route 1 (yellow). The other half of the agents obviously makes their journeys, using route 2 (green). Guidance can be given to the agents as they are about to exit link 2, that is, when they are in the blue circle. The two alternative routes are equally long, 2530 meters, resulting in travel times at free flow of approximately 91 seconds. The total travel time from the origin at link 1 to the destination at link 21 is approximately 200 seconds. The departure rate of averagely one agent per second makes the demand on link 1, 2, 20 and 21, 3600 cars/hour. Link 1 has however a capacity of only 3000 cars/hour. This non-sufficient capacity results in a situation where only 3000 cars per hour are entering the links downstream of this link. The capacity of link 2, 20 and 21 is 6000 cars/hour leading to unproblematic traffic behavior (i.e. no congestion) on and around the upstream ends of these three links. On the two alternative road stretches (link 5, 6, 14 and 7, 8, 16 respectively) the expected demand will be 1500 cars/hour. In a scenario denoted “Normal day” these demands are fulfilled since the capacity on all the links is 1900 cars/hour. In another scenario called “Accident” the capacity on link 6 (violet circle) is decreased to 1300 cars/hour, generating a congestion on route 1. It is important to note however that the total capacity on the two alternative routes, being 1300 + 1900 = 3200 cars/hour, would be large enough to handle the entire traffic demand. The accident scenario thus represents a situation where route guidance has a potential of improving the situation and it is therefore presumably a good scenario for trying out different automatic feedback control strategies. The link parameters for all links in the small test network are found in Appendix A.

The departure times of the agents in the population used for simulations on the small test network makes it reasonable to run the simulations from 07:00:00 to 09:00:00.[64] The simulations were run both in the Normal-day scenario and the Accident scenario, without as well as with the control strategies described in chapter 2.2 above. Tuning of the control parameters was consistently carried out by means of looping over a set of relevant parameter combinations and choosing the one showing the best system performance.[65] In addition to the regular control parameters, quite a few other parameters intrinsic to the COOPERS-technology also had to be tuned. These parameters are described in detail in chapter 3.2.

In addition to the two basic scenarios described above, systematic testing of the control performance in scenarios deviating from the standard ones was also carried out. In this way the robustness of the control strategies implemented was evaluated. Finally, experiments testing system performance when a normal distributed noise was added to the control output was also carried out, giving a hint on the control strategies sensitivity properties.

A more thorough description of the experiments can be found together with the simulation results in Chapter 4.1.

2.3.2 Simulations on the Reduced Berlin Network

As mentioned above, the network used for the main experiments in this master thesis is a reduced version of the full Berlin traffic network where only the major streets and minor streets connecting the major ones are included. Figure 3 shows a screenshot of a part of the network in the Berlin city centre.

[64] All or almost all cars should have left the network by nine o'clock.

[65] More systematic and sophisticated tuning procedures, such as Lambda-tuning and Ziegler-Nichol's method, were not possible to use since the system is neither linear nor unconditionally BIBO-stable. The step responses and self-oscillating system behavior needed for these tuning methods were thus not possible to induce.

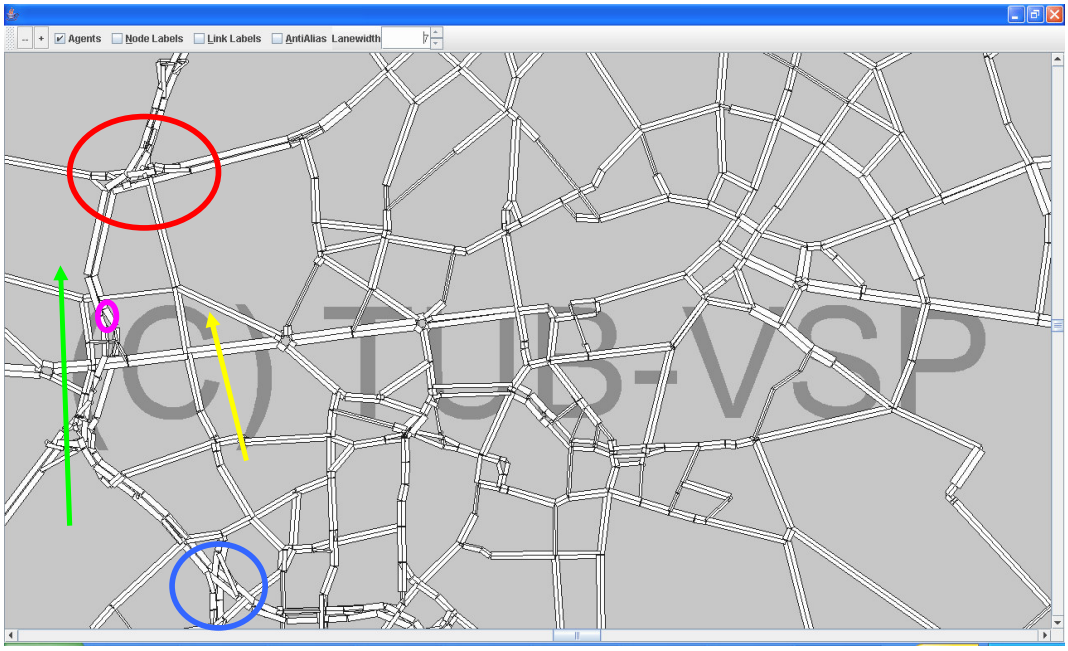

Figure 3. Central part of the reduced Berlin network. Approximately the area inside the commuter train ring-line. Guidance towards the red circle is carried out in the blue circle. Route 1 is yellow, route 2 is green. An accident is occasionally generated in the violet circle.

The simulation on this network was run with a population of some 170,000 agents. The agents' plans were, as for the full network, generated using a combination of statistical data and best path algorithm calculations. These plans specify the behavior of each agent over a 24 hour time span. My experiments did however only focus on the traffic conditions during the morning rush hours. All simulations were thus run from 5 am up until 11 am. This choice of start and end time guarantees that all interesting morning peak traffic is being considered as well as a build up phase before and a settling phase right after the rush hours.

The choice of an area where the traffic situation was to be measured and hopefully controlled was made taking several aspects into consideration. The area in Berlin chosen by the European Union as a test zone for the COOPERS-project consists of a set of freeways in the west and south-west parts of the city (highway A100, A111 and A113 respectively). In Figure 3, these freeways are the wider roads at the very left end of the figure. To set up the control on and around this area was therefore a prerequisite. The origin and destination of the guidance (blue and red circle respectively) were thereafter chosen. The choice was mainly based on the system topology. The origin and destination chosen constitute one of the few locations that have the two necessary alternative routes connecting them. Furthermore, the travel time on the alternative route (being the one marked with a yellow arrow slightly east of the highway marked with green) is only 152 second longer than the highway travel time. Guidance in this area thus has a potential of improving the traffic in situations when the traffic on the

highway is delayed more than two minutes and 32 seconds.[66] Given that the part of the highway used is the busiest road stretch in all of Germany, it is indeed realistic to believe such situations to occur at least once in a while. The two routes chosen are both approximately five kilometers long. The free flow travel time on route 1 is a little bit more than 6 minutes whereas route 2 is traversed in approximately 3 minutes and 30 seconds at free flow.

Similarly to the simulations in the small test network, a "Normal day" as well as an "Accident" scenario was run also in this larger network. In the normal scenario, the network was used unmodified whereas in the accident scenario, the capacity of link 7832 on route 2 (violet circle in Figure 3) was reduced from 2900 to 1080 cars per hour. The number of lanes was reduced as well, from three to two. This decrease of capacity represents a major accident on this link. The two scenarios were run with as well as without the different control strategies described in chapter 2.2.[67] The guidance messages were given at the end of two links leading in to the area where a route choice was to be made (Figure 4).

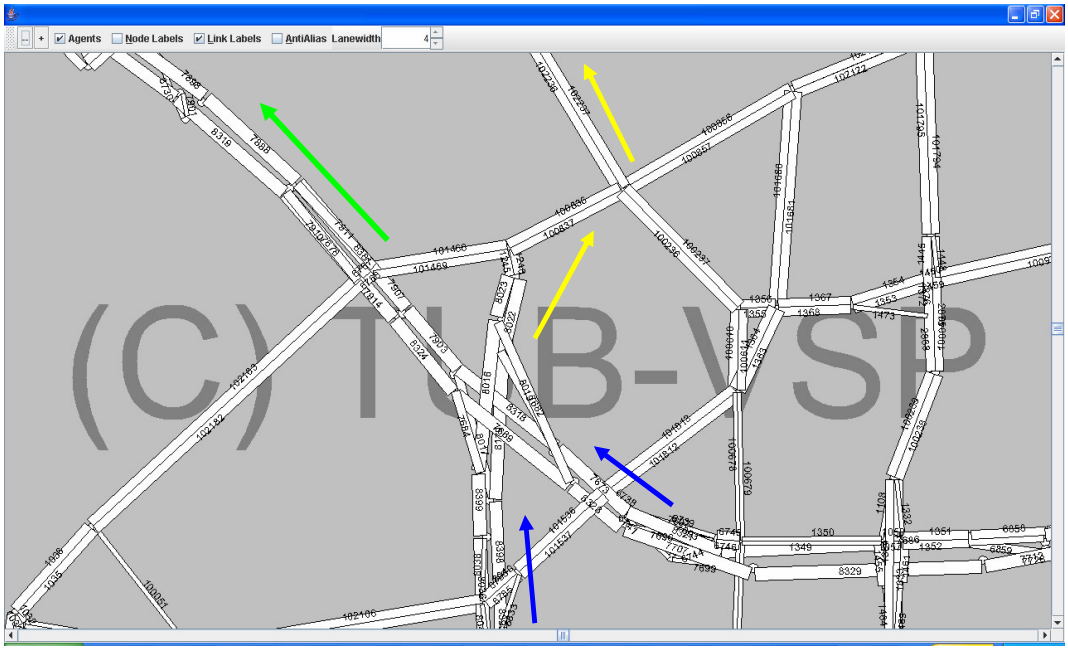

Figure 4. The links were the guidance was given to the agents (blue arrows). In Chapter 3 and the JAVA-code these links are denoted `signLinks`.

Furthermore, guidance was only given to agents planning to travel to one of the three links immediately succeeding the two routes that are controlled (Figure 5). As for the small test simulations, parameter tuning was carried out by means of trial-and-error

[66] This conclusion relies on the reasonable assumption that the entire traffic that traverse the interesting part of the system uses the highway in a normal day scenario. This is indeed the case when the plans-file is the one used for these simulations.

[67] As will be discussed in Section 4.2.1, simulations with control in the normal-day scenario, turned out to be rather irrelevant.

iterations. Due to the much more time consuming iterations (approximately 12 minutes instead of 10 seconds for each simulation), the number of iteration could not be as large as when the small network was controlled. Therefore, some of the COOPERS-technology specific parameters were not systematically optimized specifically for the large network. Instead, conclusions drawn from the small scale testing were assumed to hold even in the larger scenario. Hence, the settings that worked best for the small network were used also during the large scale simulations.

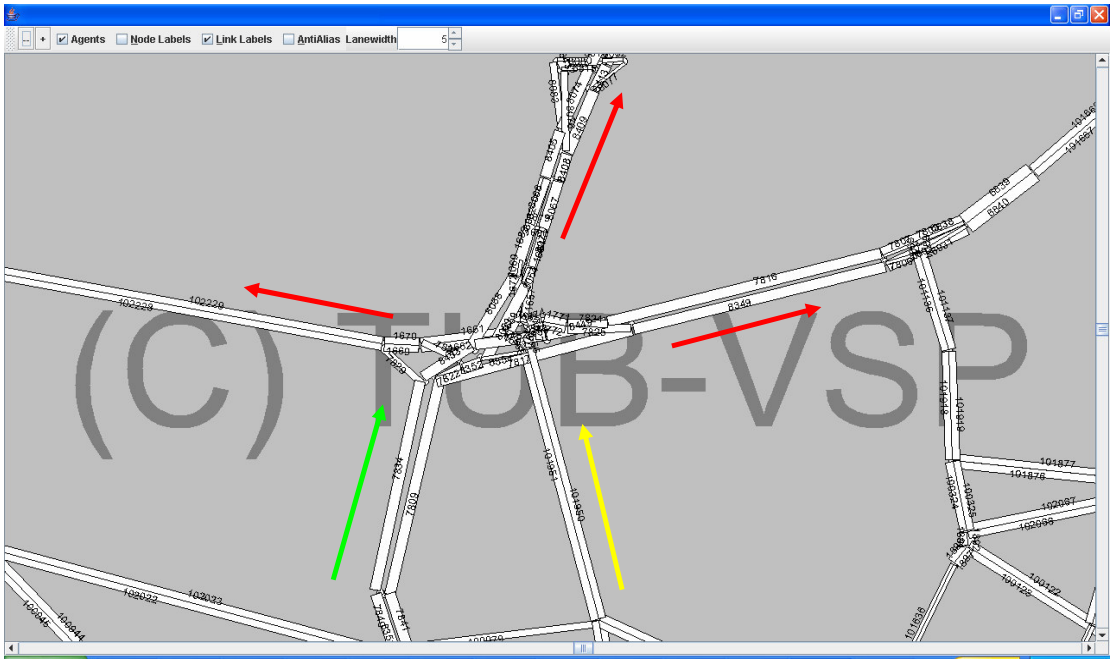

Figure 5. The links towards which the cars are guided (red arrows). In Chapter 3 and the JAVA-code these links are denoted `destinationLinks`.

Due to the much longer simulation times, robustness was not tested as systematically in the large scale scenario as in the small test scenario. A few sample experiments where the control strategies were applied with altered system settings (accident position, accident size and compliance rate) were however executed. The reason for making these experiments was not to analyze the robustness of the controllers systematically. The number of runs did not allow for such conclusions. Instead, the test experiments were meant to validate if the more systematic findings from the small test network were possible to generalize, i.e. if they were valid also in this much larger and much less controlled system.

As for the small test network, a more thorough description of the experiments being carried out in the larger network can be found together with the simulation results in Chapter 4.2.

2.3.3 Evaluation

The simulations were evaluated using several measurements. For both networks, plots of the output, $\Delta\tau_R(t)$, as well as the input, $\beta(t)$, were drawn, showing the evolvement

of the system over time.. In addition to these standard signals, an alternative output was also plotted when relevant. This alternative signal, $d(t)$ is defined as:

$$d(t) = \Delta\tau_R(t) \cdot A(t) \qquad (11)$$

In the formula, $A(t)$ represents the number of agents traveling from the common origin to the common destination routes that are using the slower of the two routes at time instance *t*. This measure gives a picture of the system behavior in which also the amount of agents *actually experiencing* a non-optimal travel time are taken into account. Hence, this signal somehow measure the total disbenefit in the system at a given time instance. The use of such a signal is most relevant in situations where all, or almost all agents, use a faster road. Looking at the normal system output $y(t)$ in such a situation might indicate that the situation is problematic since $y(t) \neq 0$. As indicated by $d(t) = 0$, the system is however in a perfect Nash Equilibrium, since no traffic is traveling on the road that is slower.[68]

In addition to plotted signals, some aggregated values describing entire simulation runs were also measured. Two of these measures correspond to the two alternative outputs described above. The first of these is a variance measure representing an average of the magnitude of $y(t)$, called Average Nash deviation (AN). The formal definition is:

$$AN = \sqrt{\frac{\sum_{t=0}^{N} y^2(t)}{N+1}} \qquad (12)$$

In this formula, it is assumed that the simulation started at *t=0* and ended at *t=N*. The unit for *AN* is seconds.

The second aggregated evaluation quantity is a similar average of the disbenefit-signal $d(t)$ discussed above. It is called Average Disbenefit (AD) and it is defined as:

$$AD = \frac{\sum_{i=0}^{N} d(t)}{N+1} \qquad (13)$$

In contrast to *AN* the output is not squared when *AD* is calculated. The reason for that is that in contrast to $y(t)$, $d(t) \geq 0$ always holds. *AD's* unit is seconds·persons.

In addition to *AN* and *AD* one more quantity was measured for each simulation. This measure, called Nash Mean (NM), was defined as follows:

$$NM = \frac{\sum_{i=0}^{N} y(t)}{N+1} \qquad (14)$$

---

[68] As will be discussed in Chapter 5.3, one could possibly argue that $d(t)$ should be used as the system output instead of $y(t)$. The reasons for not doing so were discussed in Section 2.2.1.

This measure indicates if the output has a mean that is shifted in one direction or the other. Looking at this value, one could get an indication of whether the controller produces a static error or not. Also for *NM*, the unit is seconds.

For each specific setting, final evaluating measures were achieved by taking the average values from ten simulation runs.[69] This procedure made the evaluation less dependent on the different sources of randomness inherent in the simulations.[70] When the different values were compared, the t-test was used for determining if the different settings produce significantly different values or not. The significance level of the tests was always 0.05.[71]

---

[69] For the simulations testing robustness in the large network (Section 4.2.3), only five runs for each setting were executed.

[70] The random sources in the simulations are all found in the part of the code handling guidance (see Chapter 3). That means that the averaging of several simulation runs is only necessary when guidance is applied.

[71] The t-test is defined in Råde, L. & Westergren, B., *Mathematics Handbook for Science and Engineering* 4th Edition (Lund, 2001), p 486.

3 Implementation

As mentioned already in chapter 2.1, some extra functionality had to be added to the micro-simulation software. In this chapter, the implementation of traffic measurements, COOPERS-guidance and COOPERS-perception is discussed more thoroughly. UML-diagrams complying with the standards in Martin (1997)[72] are consistently used to facilitate better understanding.

3.1 Measuring

The measuring of the system output is taken care of by a class called `NashWriter`. It is an implementation of the interface `SimStateWriterI` and as all such writers, it has a method called `dump()` which is called every second during the simulation.[73] There can be more than one `NashWriter`-object in the simulation, i.e. more than one road couple can be controlled simultaneously.[74] The basic structure of the class `NashWriter` is depicted as an UML-diagram in Figure 6.

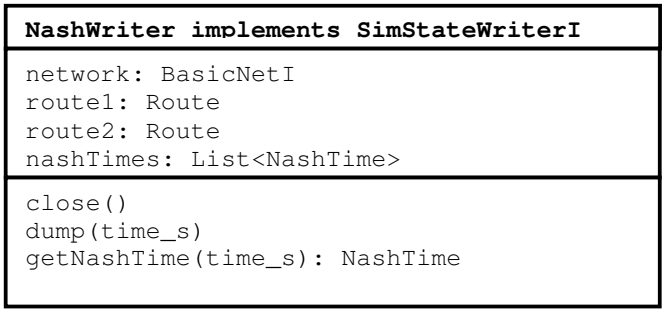

Figure 6. UML-diagram of the class `NashWriter`[75]

An object from this class knows about the network and it has two pre-specified routes, whose travel times it is to compare. The comparison is done in the dump-method and the difference between the travel times is stored in a list of `NashTime`-objects. Every such `NashTime`-object contains the time instance t given by the argument to the method, the output $\Delta\tau_R(t) = y(t)$ and the alternative control output $\Delta\tau_P(t)$. All outputs in the list are available publicly using the method `getNashTime(time_s)`.[76]

---

[72] Martin, R. C., "UML Tutorials: Part 1 – Class Diagrams" published on the Object Mentor website: http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf (2007-01-13)

[73] Writers like this one are common in MATSim. They are called every second and were originally meant to be used for simulation state documentation. In this project, the writers have more general functions.

[74] Within this master thesis project, only one road couple at a time is controlled (SISO). The code is however general enough to handle multiple control (MIMO).

[75] In this and all other UML-diagrams, only instance variables, methods, arguments and return variables relevant for a conceptual understanding are depicted.

[76] The output of the discrete system is in other words sampled at highest possible frequency. I.e. the output for every past second is available.

Right after the end of the simulation, the method `close()`[77] is called . In this method the entire output signal is written to file.[78] Furthermore, two of the aggregated evaluation values, *AN* and *NM*, are also calculated in this method and written to file. The disbenefit measure $d(t)$ and its associated aggregated value *AD* are however not calculated or written by the `NashWriter`. The reason for that is that the `NashWriter` has no access to the counting of the cars on each road.[79]

3.2 Feedback and guidance

3.2.1 `TrafficManagement` and `VDSSign`

The classes that most directly handle the guidance and feedback are called `TrafficManagement` and `VDSSign`. `TrafficManagement` implements `SimStateWriterI`. There is only one TrafficManagement-object and its main content is a list of `VDSSign`-objects. Similarly to `NashWriter`, `TrafficManagement` has a method called `dump()` that is called once every second during the simulation. This method does however nothing but calling the `dump`-methods in each `VDSSign`-object, where the real functionality is implemented. Also in the method `close()` in `TrafficManagement`, which is called after the simulation, nothing is done but calling the `close`-methods for all `VDSSign`-objects

The guidance within this project is carried out via in-car COOPERS-devices to which the guidance information is broadcasted if the car is situated on a certain link. This guidance is however fundamentally equivalent to guidance given to the cars via a variable direction sign (VDS) at the same link. This similarity explains why the JAVA-class most fundamental for the guidance is called `VDSSign`. In fact, this class does represent a VDS guiding the traffic along two routes, even though the way of transmitting the information has been modified. It is possible to have many such signs in one simulation (one for each `NashWriter`). The only difference from a situation with real physical signs is that instead of having the signs spread out in the network, they are here part of the traffic-management central, where they determine what information should be given to the COOPERS-devices. The structure of the classes involved in the guidance can be found in Figure 7.

---

[77] Also this method is a legacy method predefined by the interface `SimStateWriterI`. Originally, implementations of this interface were meant to write out the state of the simulation only. Hence, the method names `dump()` and `close()`.

[78] This data was later copied into MATLAB, where all plots were done.

[79] This quantity is instead measured and stored by the class `VDSSign`, which is discussed extensively in Chapter 3.2.

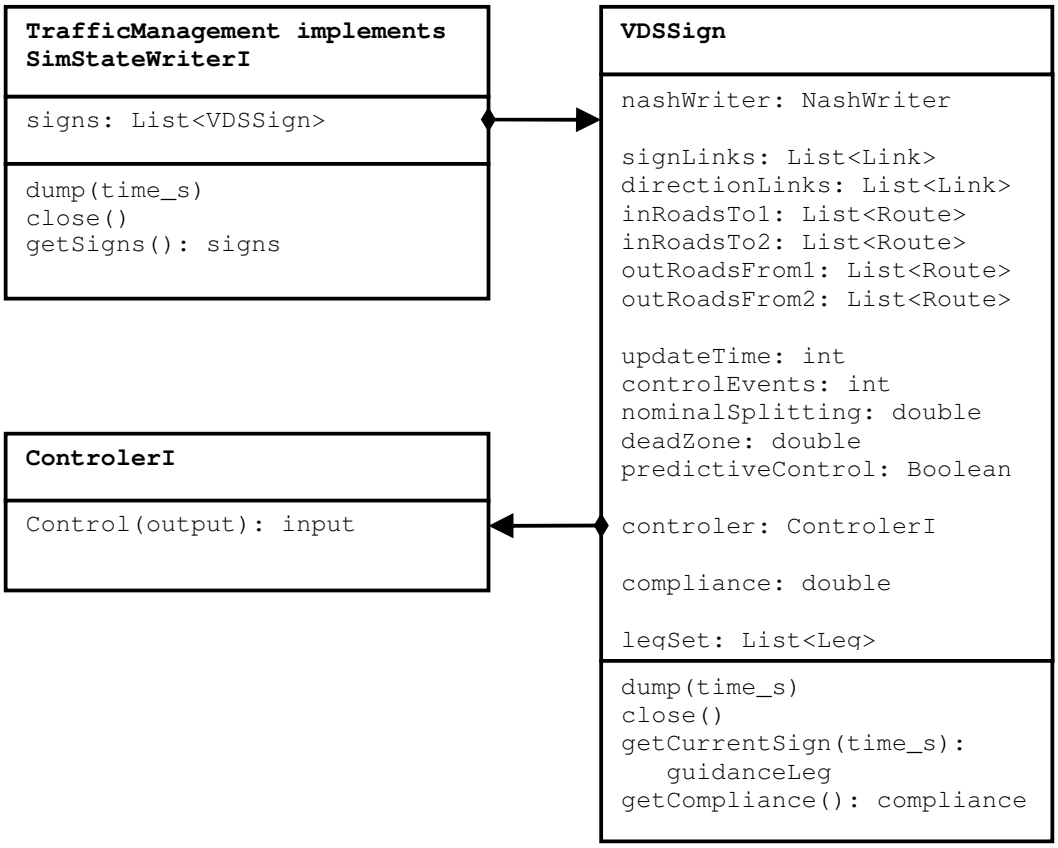

Figure 7. UML-diagram of the classes that take care of feedback and guidance.

3.2.2 Specification of relevant routes and links

A `VDSSign`-object is first and foremost defined by the two alternative routes (given by the `nashWriter`), the links where the traffic is guided, called `signLinks`[80] and the links towards which the traffic is guided, which are called `destinationLinks`. This last set of links is used to filter out the cars, which should get the information; only cars bound for these links are relevant to guide.

[80] Also the name of this variable shows the close connection between COOPERS-guidance and VDS-guidance. If the guidance would have been carried out via VDS, the signs would have been located on these links. Now, the COOPERS-devices get the information at these links.

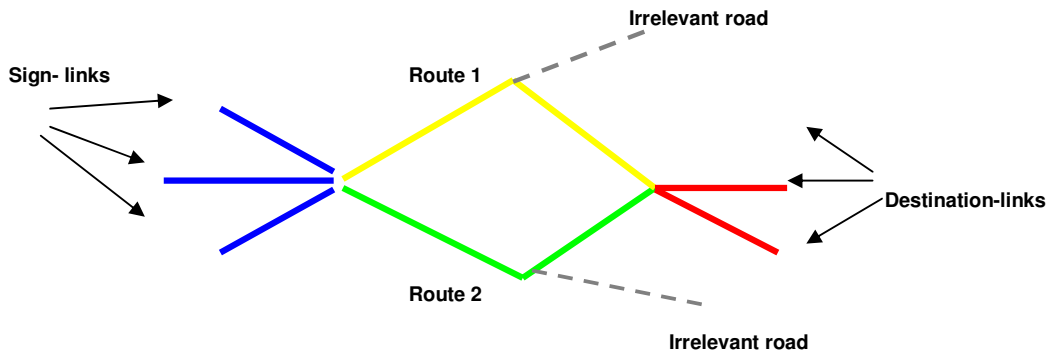

Figure 8. Cars on one of the sign-links are guided along route 1 or route 2 if they plan to pass one of the destination-links later on.

In traffic networks where the guidance is to be carried out on road stretches looking structurally like in Figure 8, the `signLinks` and the `destinationLinks` give a sufficient description of the guidance area together with the two routes in the `nashWriter`. This is often the case when the guidance is applied in inner-city networks. Unfortunately this is rarely the case on freeway-stretches such as the one controlled in the large network in this master thesis project. The reason for that is that there are rarely unifying intersections in such networks, where all `signLinks` and `destinationLinks` have direct access to the two alternative routes. Structurally, freeway networks rather look like the sketch in Figure 9 than the one in Figure 8.

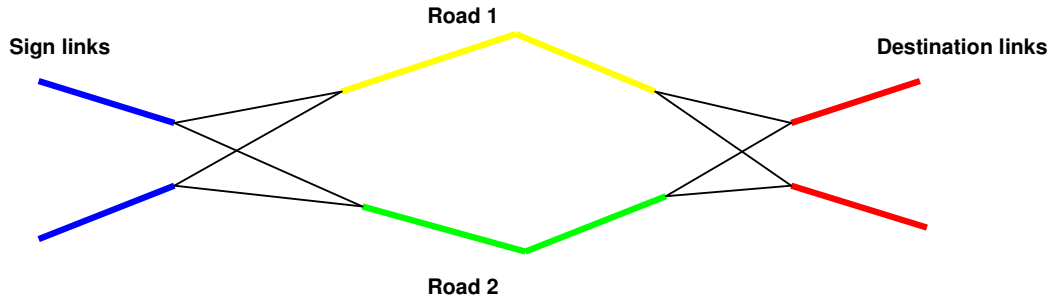

Figure 9. Typical topology of a freeway network.

The problem with this type of network is that the `signLinks`, routes and `destinationLinks` alone are not sufficient to fully describe the paths through the network. Nevertheless, such a full route is necessary if one wants to guide the traffic in a micro-simulation, since in MATSim every agent must always know its entire route to it next planned activity. A full specification of a `VDSSign`-object must therefore also contain information about how to get from the `signLinks` to the relevant route as well as from the route to the `destinationLinks`. This information is provided by the instance variables `inRoadsTo1` (how to get to route 1 from the different `signLinks`),

`inRoadsTo2` (how to get to route 2), `outRoadsFrom1` (how to get from route 1) and `outRoadsFrom2` (how to get from route 2 to the different `destinationLinks`).

3.2.3 From traffic measurements to route guidance

The core method in `VDSSign` is `dump(time_s)`, which, as mentioned above, is called every second. It has the current simulation time as its argument. In this method, the output of the system is firstly read, calling `nashWriter.getNashTime(time_s)`. Depending on the value of the Boolean instance-variable `predictiveControl`, either the predictive or the reactive travel time is registered as the system output. This output is thereafter given to the control-method in the `controler` (instance variable), which returns an input $\tilde{\beta}(t) = \tilde{u}(t) \in [-1,1]$[81]. The controller is always an implementation of the interface `ControlerI`. The algorithm in the method `control(output)` is however different in different control strategy implementations (`NoControl`, `ConstantControl BangBangControler`, `PControler` and `PIDControler`).[82]

The input $\tilde{\beta}(t)$ is then transformed to a splitting rate ($\beta(t) \in [0,1]$).[83] In this operation, a variable called `nominalSplitting`, representing u_0 in Chapter 2.2, is used. This variable represents the way the traffic is split between the two routes without any control being applied and it ensures that the controller takes this nominal splitting into consideration when setting the guidance. I.e. the control should split the traffic according to this splitting rate when the system is in Nash Equilibrium. The mapping $\tilde{\beta}(t) \rightarrow \beta(t)$ is depicted in Figure 10.

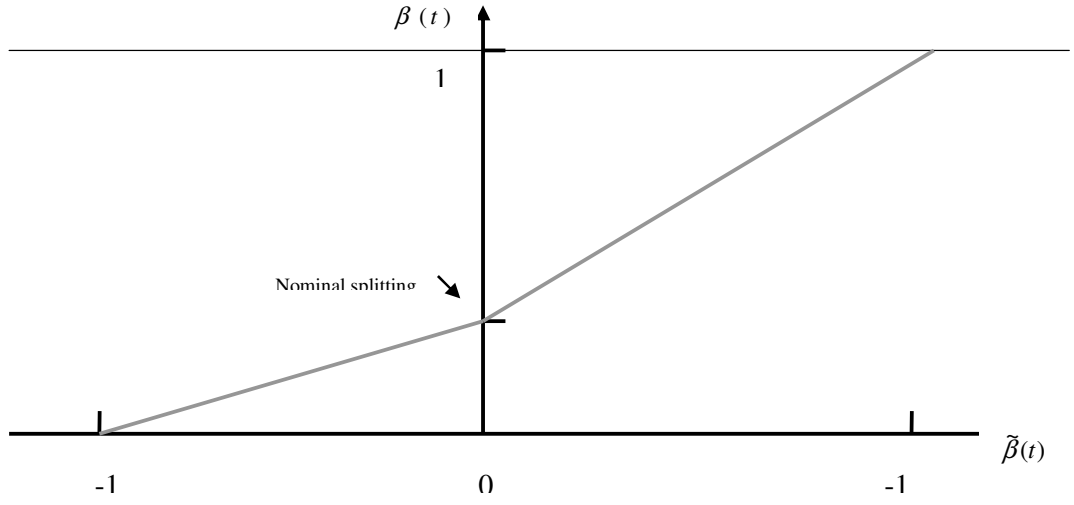

Figure 10. Transformation of the output from the controller to a splitting rate between zero and one.

[81] For the control algorithms (3) and (4) in Chapter 2.2, $\tilde{u}(t) = \hat{u}(t)$.
[82] The algorithms for the more sophisticated control strategies are described thoroughly in chapter 2.2. NoControl always returns 0. ConstantControl always returns a pre-specified value.
[83] This operation represents the last equation in algorithm, (3), (4) and (5) in Chapter 2.2.

The input $\beta(t)$ is later transformed into a sequence of guidance messages according to the strategy described in Section 2.2.6. The mapping scheme in Section 2.2.6 is however not an entirely correct description of how the algorithm in `VDSSign` actually works. In the actual implementation, there also exists a variable called `deadZone` (usually set to 0.02). If the input $\beta(t)$ is closer to the nominal splitting than this dead-zone, no guidance at all is given to the agents. This subtlety presumably makes the control less oscillating.

The mapping determines the number of two alternative guidance messages in the sequence, but the order of the messages is randomized in the `dump`-method. The guidance messages are stored as legs in a list called `legSet`. In the `VDSSign`-implementation the control parameter c and T_{fix} are represented by two `integer` instance variables called `controlEvents` and `updateTime` respectively. The `legSet`-list is normally updated every T_{hold} (`controlEvents*updateTime`) seconds. An exception is if all messages in the `legSet` are equal. In that case the sequence is updated already after T_{fix} (`updateTime`) seconds. This function speeds up the control without any loss of preciseness.

Finally, the `dump`-method also registers the current input $\beta(t)$ and the number of agents currently traveling on each road. The later of these quantities is updated by the class `COOPERSProvider`, which is discussed extensively in chapter 3.3.

3.2.4 Other methods in VDSSign

Apart from the method `dump()`, a few more methods in the class `VDSSign` are important to notice. One example is the method `close()` which is called via the `TrafficManagement` at the end of the simulation. Similarly to the `close`-method in `NashWriter`, this method writes the information collected during the simulation to file. More specifically, the input signal $\beta(t)$, the number of cars on each road, and the disbenefit values $d(t)$ are written to file in this method. Furthermore, the aggregated disbenefit value *AD* is also calculated and written down by this part of the software.

Two other important methods, used extensively by the `COOPERSProvider` discussed in chapter 3.3, are `getCompliance()` which returns the fraction of the population that actually listen to the guidance and `getCurrentSign(time_s)` which returns the current guidance, as it is defined in the `legSet`.

3.3 Perceiving and reacting

The JAVA-class handling the perception of the information broadcasted by the `VDSSign` is called `COOPERSProvider`. It implements the interface `RouteProviderI` and it is software-technically part of the agent's brain. In the real world, a `COOPERSProvider`-object can be seen as a representation of a COOPERS-device. As such, it has information of the `TrafficManagement`, which is an instance variable

called `source`, as well as of the plan of the agent driving the car.[84] A schematic UML class-hierarchy showing the entire COOPERS-guidance software and its connection to the rest of the MATSim is found in Figure 11.

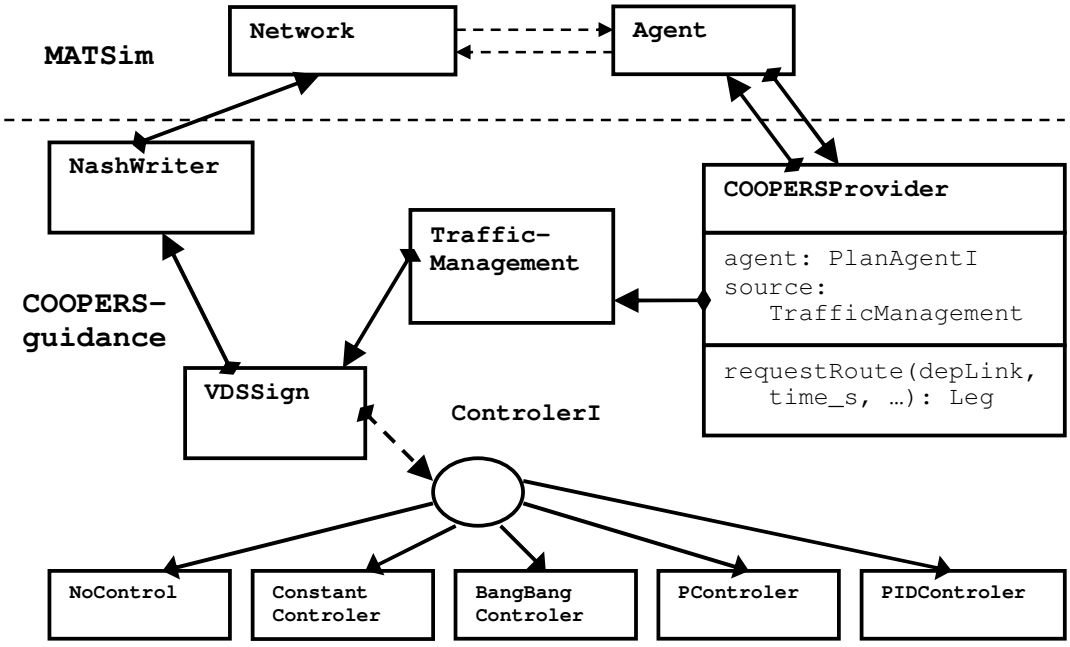

Figure 11. UML-diagram of the `COOPERSProvider` as well as its connection to other parts of the simulation.

The by far most interesting method in the `COOPERSProvider` is called `requestRoute(depLink, time_s, …)`. This method is called automatically when an agent exits a link. The method, which is specified by the interface, takes four arguments but only two are used in this implementation. These are the link about to be exited (`depLink`) and the current simulation time (`time_s`). The method returns a leg (i.e. a complete route to the next activity with a few additional properties) which replaces the current leg in the agent's plan. If `null` is returned, the agent will stick to its original plan.

The basic functionality of the `requestRoute`-method is simple. It checks if `depLink` belongs to the `signLinks` for any `VDSSign`-object in the `source` and if the agent was planning to traverse any of the corresponding `directionLinks`. In that case the agent should be guided[85] and the current guidance message is read from the `legSet` in the relevant `VDSSign`-object via the method `getCurrentSign(time_s)`. The leg returned by this method either has route 1 or route 2 as its route. Obviously, this is only a small part of the leg that `requestRoute(…)` should return. The rest of the leg is later being generated by joining appropriate links given partly by the variables in `VDSSign`

[84] As mentioned already in chapter 1.3.1.2, people using the COOPERS-devices are assumed to give the devices in their cars information about their plans.
[85] Compare chapter 3.2.2.

called `inRoadsTo1`, `inRoadsTo2`, `outRoadsFrom1` and `outRoadsFrom2` and partly by the current plan of the agent. The leg, finally returned thus gives the agent a complete route from the link where the guidance was given to the agent's destination, i.e. next activity.

In case the agent is not located on a sign-link or is not bound for any of the `destinationLinks`, `null` is returned. This is the absolutely most common scenario and it represents a situation where the `COOPERS-device` does not give the agents any guidance at all in an intersection. In such a situation, the agent sticks to its original plan. Finally `null` is also returned to a fraction of the agents for which the guidance is indeed relevant. This function is supposed to represent the fact that not all agents follow the route guidance. The fraction of the agents that do get the information when relevant is specified in the `VDSSign` as the instance variable `compliance`. This value is available for the `COOPERSProvider` via the method in the `VDSSign` called `getCompliace()`. Exactly which agents that do not comply is randomized in the `COOPERSProvider`; the compliance rate is the probability for each agent to comply. When nothing else is stated, the variable `compliance` is consistently set to 0.8 in the simulations. This choice was mainly based on the author's intuition; 80 % compliance seemed intuitively reasonable. And since this value was not regarded as totally outrageous by the traffic engineers at the department in Berlin, the intuition was gladly obeyed.

4 Simulation results

Several hundred simulations have been run during this master thesis project. Giving an account for each and every evaluation value for every single experiment is therefore not doable. Hence, it follows that the simulation runs presented in this chapter is nothing but a small selection of all simulated traffic scenarios. Only the experiments most interesting and most elucidative have been described. Furthermore, not every evaluation measurement has been explicitly mentioned for every simulation setting. As an example, the disbenefit value $d(t)$ and its aggregated counterpart *AD* are discussed more thoroughly in the section covering experiments on the larger of the two networks. The reason for that is that in the small network $y(t)$ and $d(t)$ typically show very strong resemblance. This resemblance is due to the very symmetrically distributed traffic demand between route 1 and route 2. For the large network, the demand is more unequally distributed making the $d(t)$-signal more relevant to study.

In addition to the separate simulation results discussed in the text, tables summarizing the results for all simulations in the two networks are found in Appendix B and Appendix C.

4.1 The small test network

As mentioned in Section 2.3.1, the simulations on the small test network were run in two different scenarios called “Normal Day” and “Accident”. In the following sections, the simulation results from the two scenarios are described and compared with as well as without control. The accident-case is described before the normal-day scenario. The reason for that is that the same parameters for the P- and PI-controller are used in the two scenarios. These parameters were tuned for the case “Accident” and it is therefore logical to describe this scenario prior to the normal-day case.

4.1.1 Accident case

4.1.1.1 No guidance

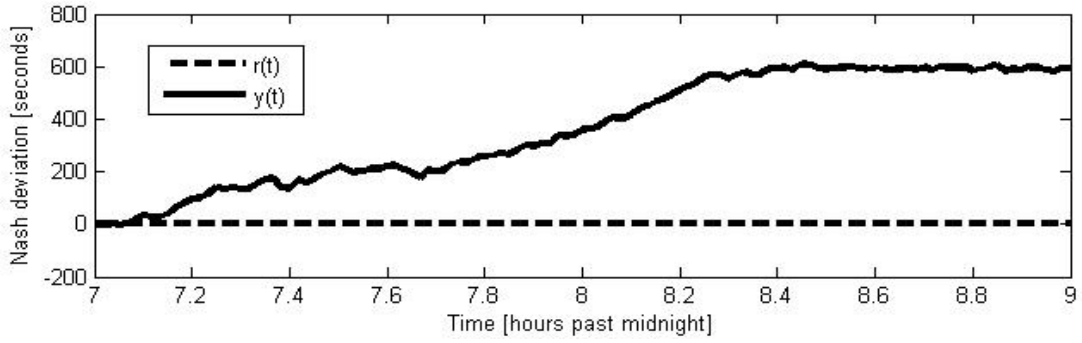

Figure 12. Output signal with accident but without control. When the output reaches 600 s, the entire route 1 is fully congested and the travel time deviation can not increase any further.

In Figure 12 one can see how the accident on route 1 makes it impossible for that route to handle its entire demand. Congestion is building up that increases the $TT_R^1(t)$ and $y(t)$ gradually. At approximately 8:15 AM the increase in $y(t)$ stops at $y(t) \approx 600$. The reason for this is that the entire route 1 upstream of the accident is fully congested. The travel time deviation can therefore not increase any further.[86] Figure 13 shows a set of snapshots from the simulation movie that shows the development of the traffic.

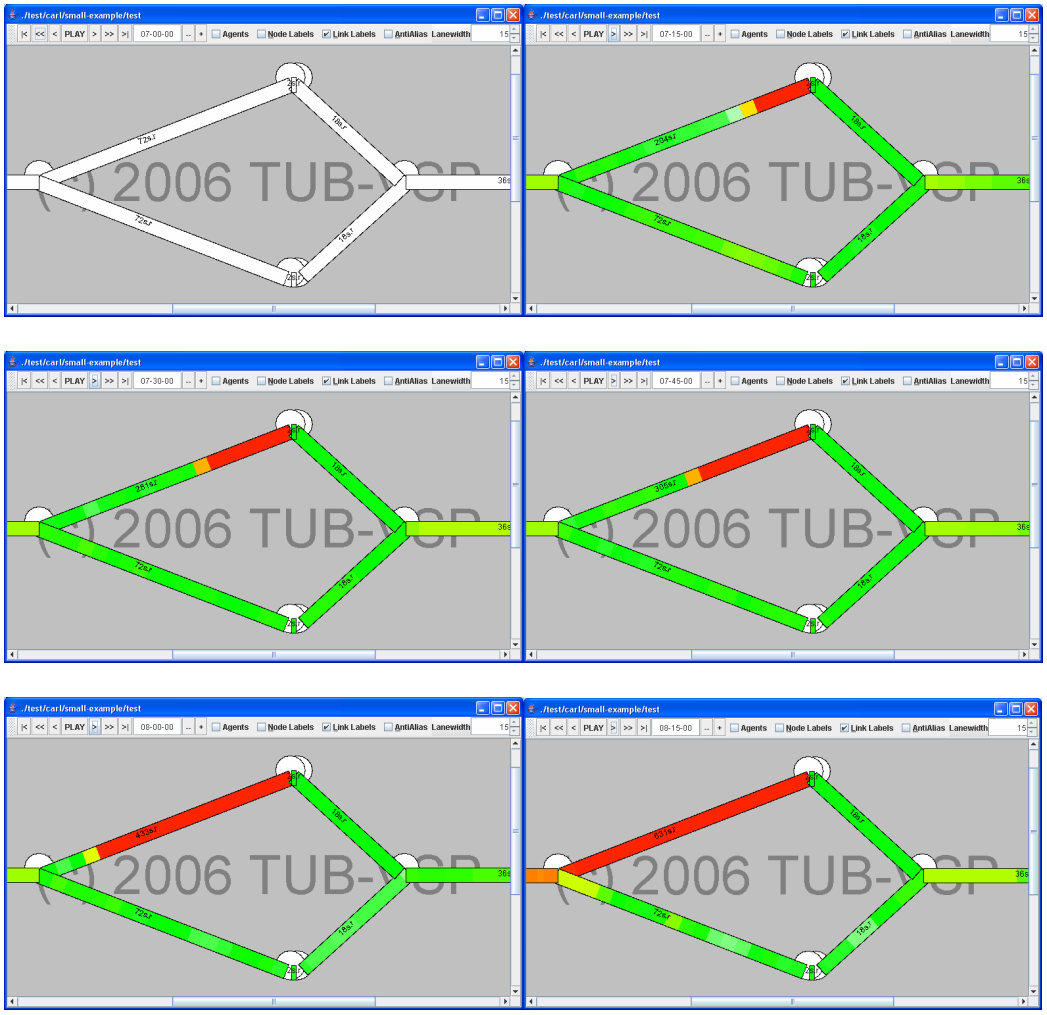

Figure 13. The development of the traffic in the small network with accident but without control. The figure shows snapshots of the movie taken at 07:00, 07:15, 07:30, 07:45, 08:00 and 08:15 AM. White color indicates no traffic, green low traffic and red represents congestion. The figure clearly shows how route 1 gets fully congested sometimes between 08:00 and 08:15 AM.

The situation in the system without control is obviously not a Nash equilibrium situation. Since no guidance is applied, 50 % of the population still uses route 1, even though its travel time is up to ten minutes longer the travel time of route 2. The

[86] The spill-back queue upstream of the node where the traffic split does indeed increase the travel times on these links. But since these links are part of both route 1 and route 2, the travel time *difference* will not increase.

Average Nash-value (*AN*) is 419 s, the Nash mean (*NM*) is 364 s and the average disbenefit-value (*AD*) is 80,354 s·person.[87]

4.1.1.2 Static guidance around the accident

One obvious solution to the problematic situation generated by an accident like this would be to guide all traffic along the alternative route 2. Such a static guidance seems reasonable to apply if no dynamic guidance (VDS, COOPERS etc.) is available. A temporary static signpost is then placed in the intersection. Nevertheless, as is shown in Figure 14, guiding all agents into route 2 does not solve the problem.

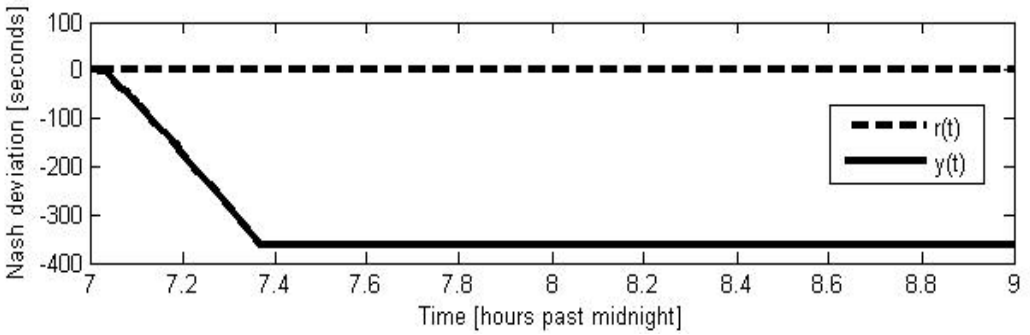

Figure 14. Output with accident when all traffic is guided into route 2.

The reason for this is obviously that the capacity of route 2 is not large enough to handle the entire demand.[88] The guidance only moves the congestion problems from route 1 to route 2. The traffic situation does get a bit better than without guidance, since when route 2 is fully congested, the travel time difference is not as big as before ($y(t) \approx -370$ at steady-state). Also *AN*=337 and *AD*=75384 indicate a slight improvement. The input signal, shown is Figure 15, is obviously $u(t) \equiv 0$.

[87] As mentioned in chapter 2.3.3, aggregated measures are normally averages of 10 diffent runs (although in this particular case, averaging is not really needed since all random sources are in the guidance-code). For a few experiments, covered below, only five runs were averaged. When that is the case, it will be clearly mentioned. Otherwise, all measures are averages of ten runs.

[88] One should remember however, that guiding all traffic into route 2 does not mean that *all traffic is actually using* that route. The reason for this is of course that the compliance rate is not 100 but 80 % in this as well as all other simulations. Hence, 20 % of the travellers do not care about the guidance. Out of these agents, 50 % are using route 1. In total, 90 % of the cars are thus using route 2 whereas 10 % travel on route 1.

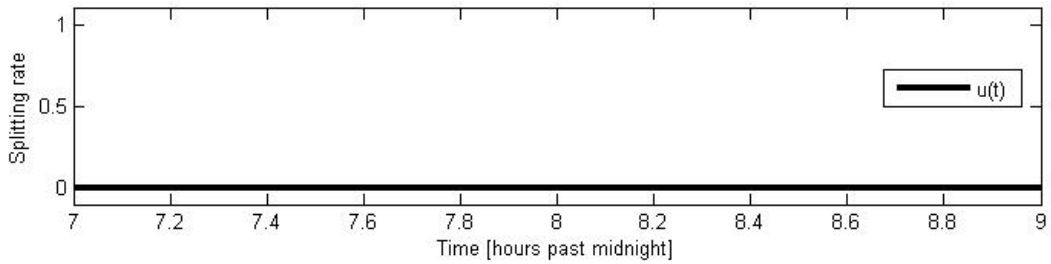

Figure 15. Input signal when all cars are guided into route 2.

The plot in figure 14 and the associated evaluation measures, clearly show that static guidance can not handle an accident situation like the one simulated. The evaluation measures did certainly improve when the guidance was installed, but the situation is still not at all satisfactory.

4.1.1.3 Bang-bang control

The most trivial way to guide the traffic dynamically is by implementation of the bang-bang control strategy. As clearly depicted in Figure 16, this control (based on $\Delta\tau_R(t)$) improves the system performance considerably. Although the system oscillates strongly, $|y(t)| < 200$ always holds. *AN*=112 s is also significantly better than before (approximately 26 % of the same value without guidance) and *NM*=39.1 s, is closer to zero. The largest improvement can however be observed at *AD*=6660. This value is only 8.3 % of the same value without guidance. The reason for this is obviously that with bang-bang control most agents are actually using the fastest road whereas before, 50 % of the agents used the slower of the two roads.[89]

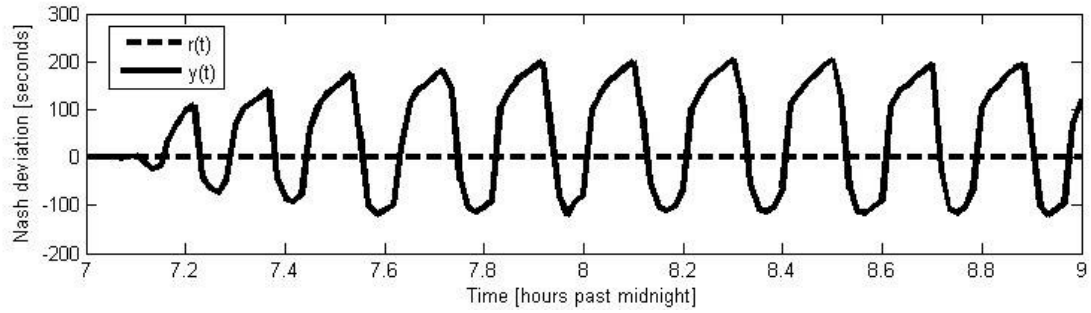

Figure 16. Output with accident and bang-bang control based on reactive system output. The oscilating system behavior is expected for bang-bang controlled systems.

The input generated by the bang-bang controller is shown in Figure 17. In the figure one can easily see that the control algorithm (2) from chapter 2.2.2 is followed correctly. That is, the input equals one when the output is negative, zero when the output is positive and 0.5 initially as the output is equally zero.

[89] For the static guidance, it was even worse; i.e. 90 % of the agents were then utilizing the slower road.

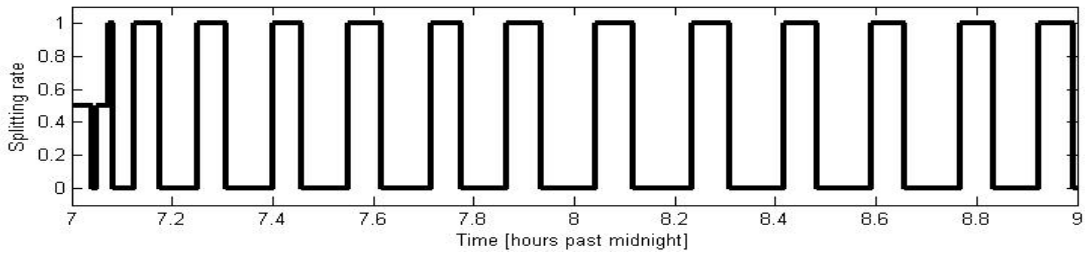

Figure 17. Input from the Bang-Bang controller when reactive travel times are used.

As discussed in Chapter 2.2.2, the oscillating system behavior experienced is something one would expect when bang-bang control is applied. The reason for that is that the bang-bang controller tends to do too much for too long time. In general, bang-bang control is sensitive to time lags and the oscillations are usually damped considerably if the time lags are reduced. One would therefore expect predictive measurements to be favorable when the system is controlled by a bang-bang controller.[90]

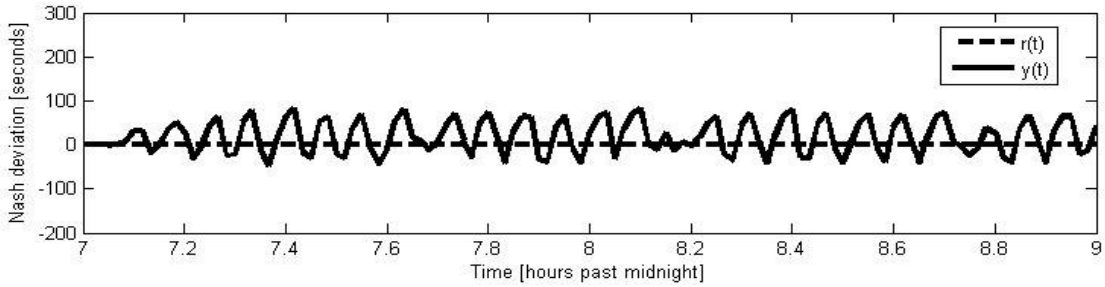

Figure 18. System output with accident and bang-bang control based on predictive travel times.

Figure 18 shows that assumption to be valid. The magnitude of the oscillations gets much smaller ($|y(t)| < 100$) when the controller uses $\Delta\tau_P(t)$ instead of $\Delta\tau_R(t)$ as its control output. With predictive measurements, the controller quicker understands that the output is about to change sign. Hence, the frequency of the oscillations increases. The increase of the frequency can also be observed for the control signal, shown in Figure 19.

[90] The relation between predictive control and time lags is discussed in chapter 2.2.5.

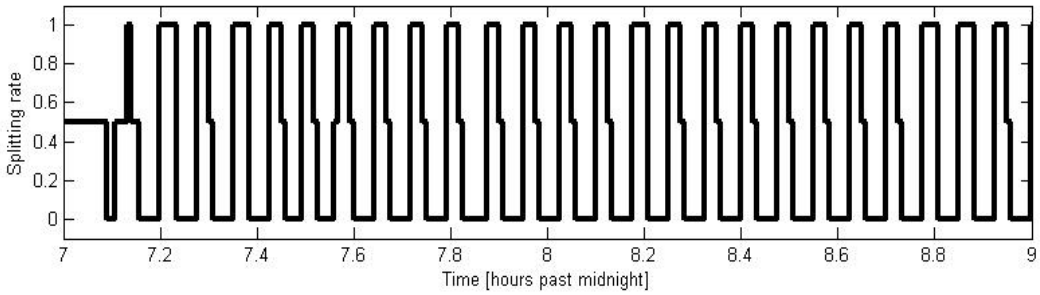

Figure 19. Input from the bang-bang controler when predictive travel times are used.

Looking at the three aggregated evaluation measures, one can once again conclude that the bang-bang control works better with predictive system output. *AN*=45.1 s (112 with reactive output), *NM*=18.8 s (39.1) and *AD*=1363 s·person (6660). The improvements of all three values are significant.

4.1.1.4 Proportional control

4.1.1.4.1 Parameter tuning

In contrast to bang-bang control, proportional control can not be implemented without a certain amount of parameter tuning. Apart from the control parameter intrinsic to all P-control (K_P), some parameters associated with the implementation of `VDSSign` also needed to be specified.[91] These parameters were all specified for a controller basing the control output on reactive travel times. The parameter among the `VDSSign`-variables easiest to define for the small test network was the one called `nominalSplitting` (i.e. u_0). The traffic flow is known to be split equally between the two roads and `nominalSplitting=0.5` was accordingly an obvious choice.

Another parameter necessary to specify was the `VDSSign`-variable called `deadZone`. The specification of this parameter was, admittedly, not done in the most systematic way. The reason for this was that it would have been extremely time consuming to test all possible combinations of all parameters that one is allowed to choose freely. Some parameters, considered to be less essential for the overall system performance, were thus chosen independently of the other parameters. Thereafter, other more important parameters were tuned systematically with the less important parameters kept constant.

The choice of dead-zone was done by first choosing $K_P = 0.001$ and `controlEvents=3`. Picking $K_P = 0.001$ seemed intuitively reasonable since it implies that $\hat{u}(t) = -K_P \cdot y(t)$ gets a reasonable magnitude[92] and, as will be described later in this section, `controlEvents=3` turns out to work fine. Thereafter, simulations were run with a few different dead-zone values. The results of these simulations are depicted in Figure 20. The results are unfortunately not very clear, but one conclusion

---

[91] The `VDSSign`-parameter `updateTime` is assumed to be 30 seconds throughout the entire project. Hence, this variable is not discussed in this section.

[92] $y(t) \approx 600$, as in figure 12, implies that $\hat{u}(t) = -0.6 \in [-1,1]$.

that can be drawn is that it is always better to have a dead-zone that not having one.[93] The value finally chosen was 0.02. The *AN*-value generated with this dead-zone was the best achieved. It was significantly better than the value achieved without dead-zone.[94]

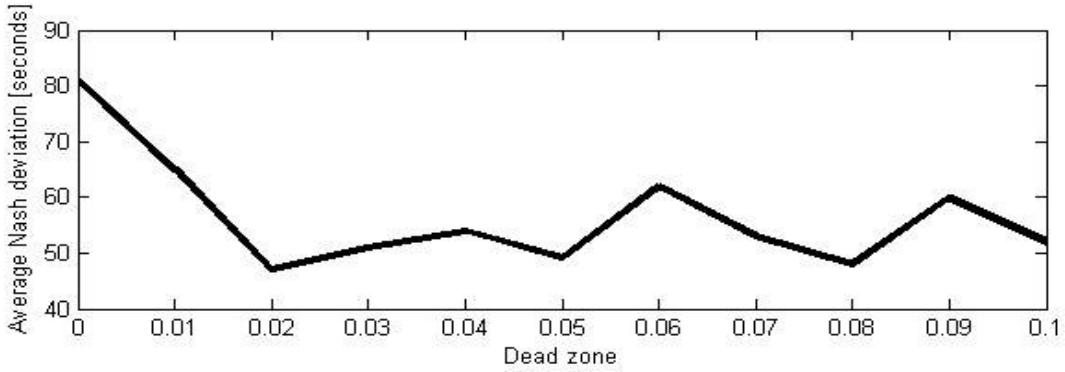

Figure 20. Plot showing AN as a function of the dead-zone with P-control when Kp=0.001 and `controlEvents=3`*. The AN-values are averages from ten runs.*

Having fixated the dead-zone, the `VDSSign`-parameter denoted `controlEvents` was the next to be chosen. That this parameter has a potential of affecting the control performance heavily was discussed already in Chapter 2.2.6. This time, $K_P = 10^{-3.5} \approx 0.00032$ was chosen.[95] Simulations were run with the `controlEvents`-variable taking all whole-number values between one and ten. For each setting, ten runs were carried out and the *AN*-values were averaged. Figure 21 shows the results.

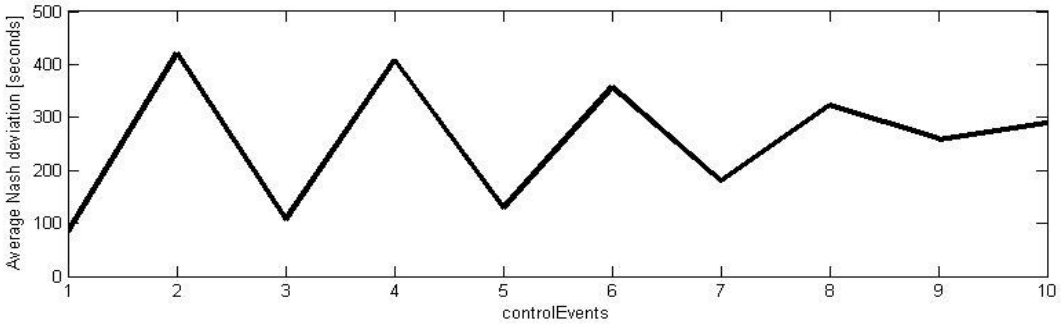

Figure 21. Plot showing AN for P-control (Kp=0.00032) and different `controlEvents`*-values.*

The most obvious conclusion drawn from Figure 21 is that `controlEvents` should definitely be chosen odd-valued. In fact, if one use for example `controlEvents=2` one could as well use no control what so ever. As an example, `controlEvents=2`

---

[93] Even when the P-parameter was optimized with `deadZone=0`, *AN*=44.1 and *AD*=1675 were the best evaluation measured obtained. As is shown later on, this is clearly worse than the best values obtain with P-control and `deadZone=0.02`. (The difference is significant)

[94] This *AN*-value was however not significantly lower than for many other non-zero dead-zones.

[95] Admittedly, it would have been more logic to choose 0.001 also this time. There was no really good reason for changing the control parameter, even though 0.00032 also generates inputs of reasonable magnitude.

gives $AN \approx 400$; without control $AN=419$ is achieved. Other even numbers are almost as bad. This might look weird, but the explanation of this phenomenon is actually rather straightforward. If `controlEvents=2`, the mapping $\beta(t) \rightarrow \beta_{real}(t)$ looks like this:

$$\begin{aligned} &\beta(t) = [0,0.33) \rightarrow \beta_{real}(t) = 0 \\ &\beta(t) = [0.33,0.67) \rightarrow \beta_{real}(t) = 0.5 \\ &\beta(t) = [0.67,1] \rightarrow \beta_{real}(t) = 1 \end{aligned} \qquad (15)$$

In contrast to the corresponding mapping when `controlEvents` is odd (9), the mapping in (15) does not give different $\beta_{real}(t)$ for $\beta(t)$-values slightly below and slightly above 0.5. Contrary, $\beta_{real}(t) = 0.5$ for the entire interval $\beta(t) \in [0.33,0.67)$.[96] Noticing that $\beta_{real}(t) = 0.5$ implies that the traffic is split equally between the two routes[97], one understands why a P-controller with `controlEvents=2` resembles no guidance at all in practice. Only $\Delta\tau(t)$-values with very large absolute values induce $\beta(t)$-values that are large enough to avoid $\beta_{real}(t) = 0.5$. The same argument also holds for other even `controlEvents`, although since the $\beta(t)$-interval generating $\beta_{real}(t) = 0.5$ gets smaller, the control does not get all that bad.

When `controlEvents` instead is chosen odd, $\beta_{real}(t)$ will always take different values for $\beta(t)$-values slightly below and slightly above the dead-zone surrounding 0.5. That is, $\beta_{real}(t)$ is never 0.5 in such a setting.[98] The controller will therefore be much more active; in other words, it will actually control the traffic more frequently.

Looking back at Figure 21, `controlEvents=1` generates the very best results. Choosing this parameter value would however transform the P-controller into a bang-bang controller (as discussed in chapter 2.2.6). Nonetheless, such a controller would be totally insensitive to variations in the K_P-value. The `controlEvents`-parameter was therefore finally set to 3. As is shown in Figure 21, this setting produced the second best evaluation-measures[99] yet one can assume this setting to generate better results than `controlEvents=1` after the K_P-parameter has been tuned.[100]

The tuning of the control parameter K_P was done by means of testing a set of reasonable candidates. Similarly to when `controlEvents` was chosen, ten simulations were run for each K_P-value, the evaluation measure *AN* being an average from these runs. A first rough test series produced the plot in Figure 22.

---

[96] If $\beta(t)$ is in the dead-zone, `null` is returned.
[97] Over a one minute time interval.
[98] Although, it can be that there is no control at all if $\beta(t)$ is in the dead-zone.
[99] The differences between the *AN*-values achieved with `controlEvents=1` and `controlEvents=3` were by the way not significant.
[100] As will be shown later on in this section, the P-controller with `controlEvents=3` did generate better system performances than the bang-bang controller. Hence, `controlEvents=3` was a good choice.

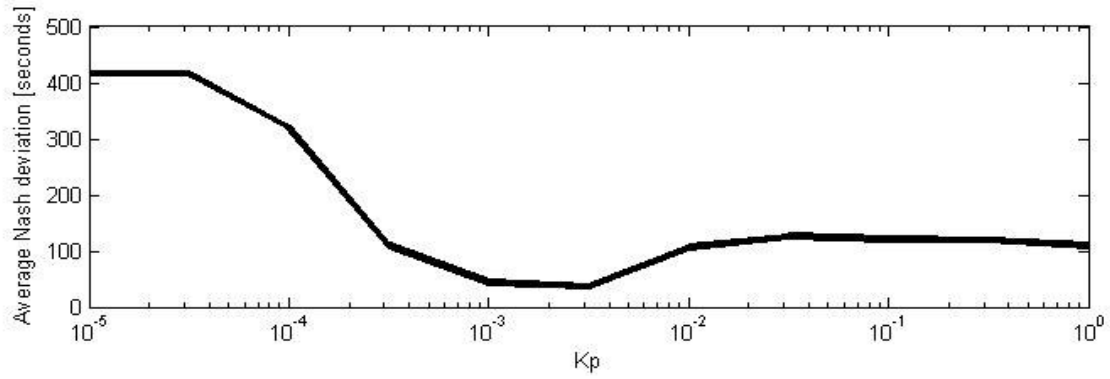

Figure 22. Rough tuning plot for the P-controller's control parameter K_P.

The plot clearly shows that neither a too large nor a too small K_P gives good control performance. When K_P is too small, there is no control at all resulting in *AN*-values similar to those given in the simulations without guidance. With too large K_P, the P-controller acts like a bang-bang controller. As expected, the *AN*-values resemble the value obtained with a reactive bang-bang controller (112 s) when K_P is really high.

The optimal K_P apparently lies close to 0.001. Another set of simulations, with $K_P \in [10^{-3.2}, 10^{-2.2}]$, was therefore run to make the tuning more exact. Figure 23 shows that the very best K_P-value turned out to be $10^{-2.8} \approx 0.0016$. This value was therefore chosen.

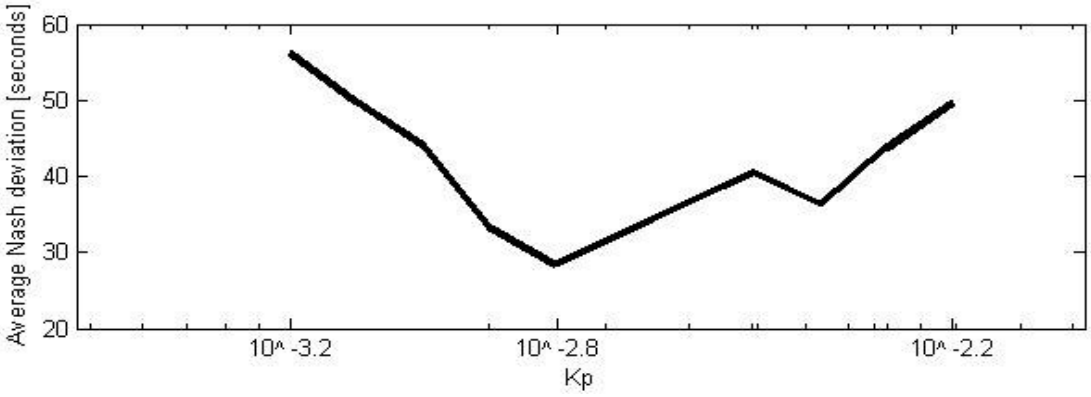

Figure 23. Fine tuning plot for the P-controller's control parameter.

4.1.1.4.2 Simulation runs

Controlling the accident scenario with a P-controller using the parameter setting described in the preceding section and reactive travel times, typically generates outputs and inputs as in Figure 24 and 25.

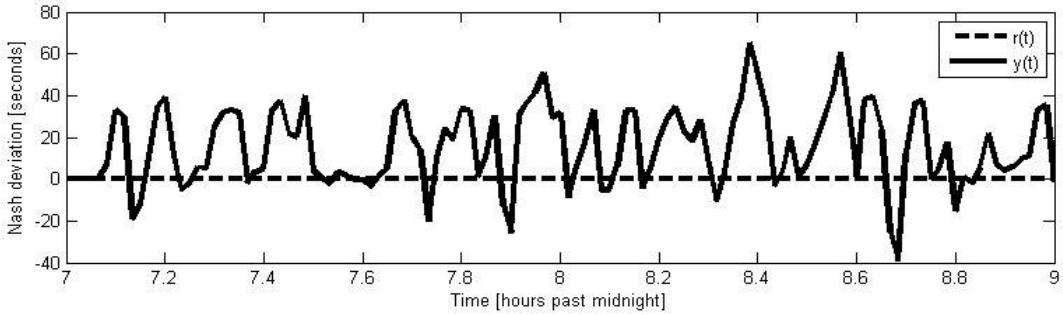

Figure 24. Output with accident when the system is controlled by an optimally tuned P-controller based on reactive travel times.

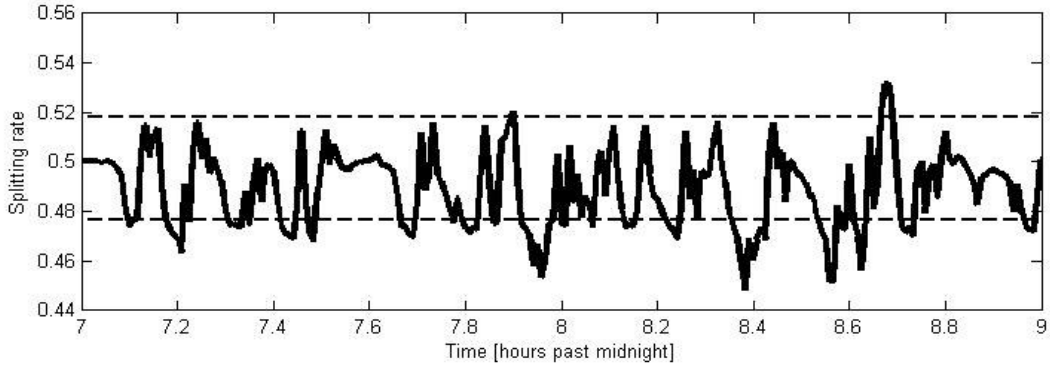

Figure 25. Input to the system when it is controlled by an optimally tuned P-controller based on reactive travel times. The area between the two broken lines is the dead-zone.

The output plot alone clearly shows that an optimized P-controller works better than the bang-bang controller discussed in chapter 4.1.1.3. The output only rarely exceeds 60 s and it never goes below -40 s. The aggregated evaluation measures tell a similar story; *AN*=28.0 s and *AD*=927 s·person are both significantly lower than the corresponding values for the bang-bang controller, even when predictive travel times were used.[101] The third aggregated quantity measured, *NM*, did however not improve significantly. With predictive bang-bang control, this value was 18.8; with reactive P-control it was 16.5. This observation should be interpreted as signifying that the P-control is not able to eliminate static errors. In steady state, the system therefore oscillates back and forward around a non-zero level. Another interesting aspect of the P-controlled simulation is that the input produced by the controller only rarely goes outside the dead-zone area (Figure 25). This implies that the controller does not control the system at all most of the time (i.e. no guidance is shown by the COOPERS-devices). It is interesting to observe that a controller can be so very efficient, and indeed optimized, without actually doing anything at all most of the time.

[101] With predictive bang-bang control, these two evaluation measures were 45.1 s and 1363 s·persons respectively.

Also for proportional control, a set of simulations were carried out using predictive instead of reactive travel times. The results turned out to be more or less identical to the results obtained when reactive travel times were used. No significant differences at all were observed and the plots are therefore omitted in this report.[102]

4.1.1.5 Proportional and integral control

4.1.1.5.1 Parameter tuning

Seeing that `deadZone=0.02` and `controlEvents=3` worked out well for the P-controller, these values were used also for the proportional an integral controller. The main reason for not re-tuning these parameters was that it would need another set of costly simulation series. Nonetheless, seeing that these two parameters are not controller specific but rather something involved in the input actuation, it is reasonable to assume that parameter values that worked well for P-control would work well also when an integral part was added to the controller.

The tuning of the two parameters K_P and T_i was done by testing a set of parameter combinations that seemed reasonable. As a rule of thumb, the optimal K_P-value in a PI-controller tend to be approximately 10% smaller than the K_P in an optimal P-controller for the same system.[103] As described in Section 4.1.1.4.1, the optimal K_P-value for the P-controller was found to be $10^{-2.8} \approx 0.0016$. As a first guess, $K_P = 10^{-2.8} \cdot 0.9 \approx 10^{-2.9}$ thus seemed like a reasonable choice. The nine K_P-values considered in the tuning process were equally distributed (logarithmically) around this first guess. In order to find an appropriate set of T_i-values for the tuning, some PI-controlled simulations with $K_P = 10^{-2.9}$ and randomly picked T_i-values were run. It was shown that $T_i \in [100,5000]$ is an interval that certainly includes the best T_i. Five T_i-values in this interval ($T_i \in \{10^{6/3}, 10^{7/3}, 10^{8/3}, 10^{9/3}, 10^{10/3}\}$) were thus used in the tuning. The tuning results, showing the averaged *AN* –values for simulations run with all relevant K_P-T_i-combinations, are shown in Figure 26.

[102] The aggregated evaluation measures form these experiments are found in Appendix B.
[103] This is rule of thumb is part of the Ziegler-Nichol's method for tuning PID-controllers. The method is described in Glad & Ljung (1989), p 48.

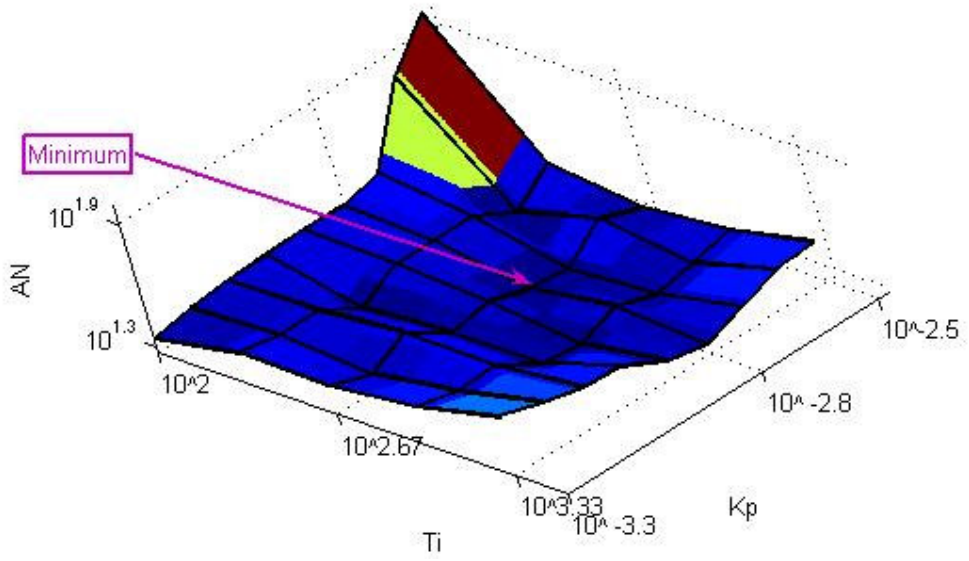

Figure 26. Average Nash deviation as a function of K_P *and* T_i*. Dark blue color indicates that AN is low. The minimal AN-value is achieved when* $K_p = 10^{-2.8}$ *and* $T_i = 10^{8/3} = 10^{2.67}$.

It is not totally trivial to interpret the tuning results from Figure 26 alone. One can however easily see that *AN* grows really big when both K_P and $K_i = 1/T_i$ are large. In that case, the controller resembles a bang-bang controller, which is seen on the *AN*-value that tends towards 100 s. The corresponding value for reactive bang-bang control is, as mentioned in Section 4.1.1.3, *AN*=112 seconds.

One other conclusion that can be drawn is that a large number of parameter combinations give approximately the same *AN*-values (between 20 and 30 seconds). A minimum can however be found at $K_P = 10^{-2.8}$ and $T_i = 10^{2.67}$. These parameters were hence considered to be optimal and in the coming sections, all simulations are run with this parameter setting.

4.1.1.5.2 Simulation runs

Controlling the accident scenario with a PI-controller using the parameter setting described in the preceding section and reactive travel times, typically generates outputs and inputs as in Figure 27 and 28.

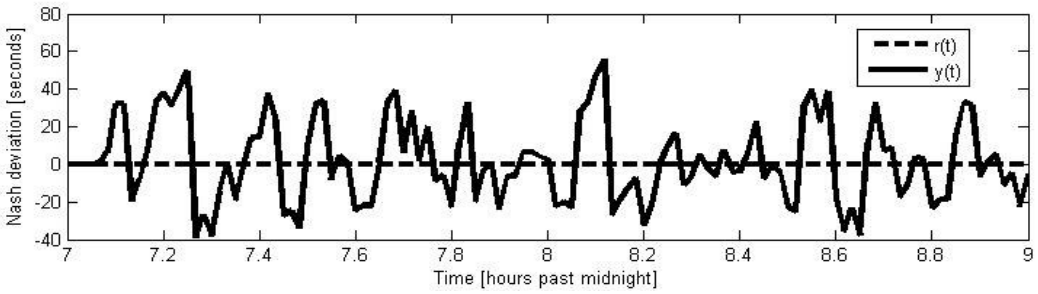

Figure 27. Output with accident when the system is controlled by an optimally tuned PI-controller based on reactive travel times.

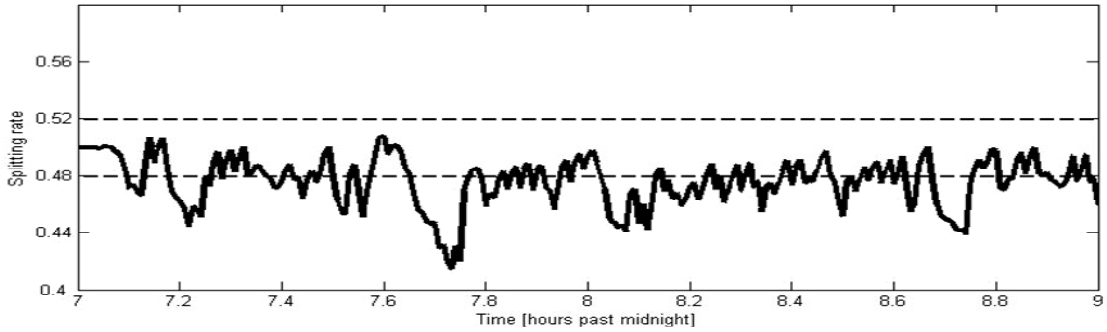

Figure 28. Input to the system when it is controlled by an optimally tuned PI-controller based on reactive travel times. The area between the two broken lines is the dead-zone.

The output in Figure 27 is apparently better than the corresponding output when P-control was applied (Figure 24). This time $y(t)$ only rarely exceeds 40 seconds. The aggregated evaluation values are also better with PI-control than with P-control. *AN*=22.2 s, *AD*=745 s·person and *NM*=2.07 s. All these measures are significantly lower than the corresponding values for P-control. Especially interesting is the improvement of *NM*. As mentioned in Section 4.1.1.4.2, the proportional controller did not manage to improve the shifted mean in the output, which was almost as big as for the bang-bang controller (16.5 s). That was interpreted as a sign of the P-controller's inability to avoid static errors. Applying PI-control to the system, the shifted mean almost disappears completely (*NM*=2.07 s); in other words, the integration eliminates the static error as expected.

Finally, one more detail to notice is that the input from the PI-controller (Figure 24) is not in the dead-zone as often as the P-controller's input (Figure 25). The reason for this is that the entire $u(t)$ is slightly shifted downwards. The periods when the input is in the dead-zone therefore gets shorter and rarer.

Also for the PI-control, a set of simulations were carried out using predictive instead of reactive travel times. The results turned out to be more or less identical as when

reactive travel times were used. No significant differences at all were observed and the plots are therefore omitted in this report.[104]

4.1.2 Normal day

4.1.2.1 No guidance

The output from the system during a normal day without accident typically looks like the plot in Figure 29.

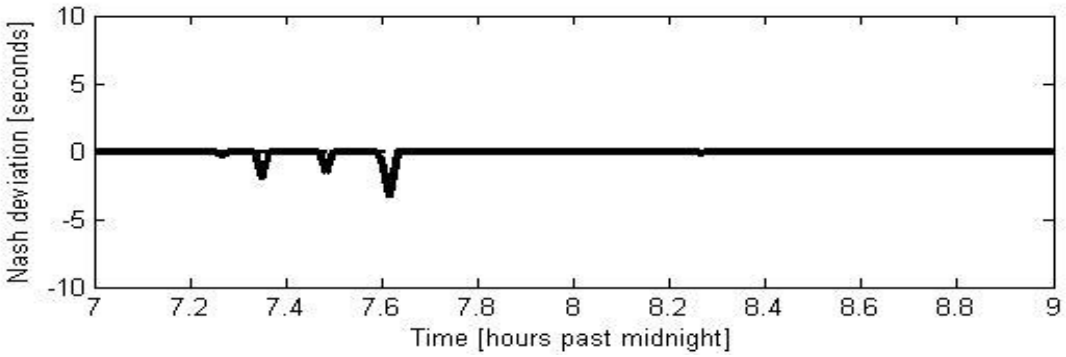

Figure 29. Output on a normal with neither accident nor control.

The situation is almost perfect. *AN*=0.423, *NM*=-0.061 and *AD*=3.5.

Automatic feedback controllers are supposed to be flexible enough to be turned on at all times. I.e. it is a necessity that the controller does not make the system performance worse in situations, like the one in Figure 29, where no control is needed. For all controllers used in this master thesis, it holds that $y(t) \equiv 0 \Rightarrow u(t) \equiv u_0$. Due to the fact that the controller has a dead-zone, this means that no guidance at all is given in this case. Hence, no controller used in this project runs the risk of distorting a system that works fine also without control.[105]

The conclusion that no controller could ever destroy a Nash equilibrium situation is however only valid if $y(t) \equiv 0$ holds exactly. As one can see in Figure 29, that is not exactly the case. In the real world, one can also assume that a lot of disturbances affect the output so that $y(t) \equiv 0$ never holds. In order to test the controllers' abilities to keep a Nash equilibrium in such noisy environments, a normally distributed disturbance with a standard deviation of 5 seconds, was added to the output. Without control, the output looked like in Figure 30 when such a disturbance was added. This output is basically nothing but a series of random numbers with an expectation of 0 and a standard deviation of 5. Non-surprisingly, *AN*=5.03 and *NM*=0.0805 in this simulation. *AD* is 149 s·person.

---

[104] The aggregated evaluation measures from this experiment are found in Appendix B.

[105] Simulations of normal-day-scenarios controlled by all controllers discussed in Chapter 4.1.1 were run to test this intuitive hypothesis. No result significantly different from the results without control (Figure 29) was observed.

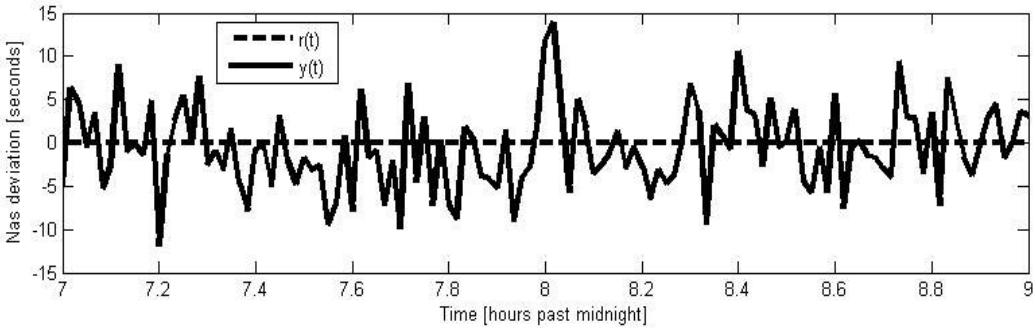

Figure 30. Output on a normal day when a normally distributed disturbance with a standard deviation of five is added to the output.

4.1.2.2 Guidance with disturbed output

When bang-bang guidance is applied in the system with disturbed output, the situation is considerably worsened. The guidance, trying to compensate for the disturbances, quickly controls the system too much, generating oscillations. As for the accident scenario (Section 4.1.1.3), the oscillations get smaller when the control output is based on predictive instead of reactive travel times. Figure 31 and 32 shows the output from systems controlled with reactive and predictive bang-bang-control respectively. Also the aggregated evaluation measures obviously get larger than without control. As an example, *AN*=33.3 when the control is reactive. That is almost 7 times worse as without control. When the control is predictive, *AN*=19.4 which is better but still almost four times as bad as when no control at all was applied.

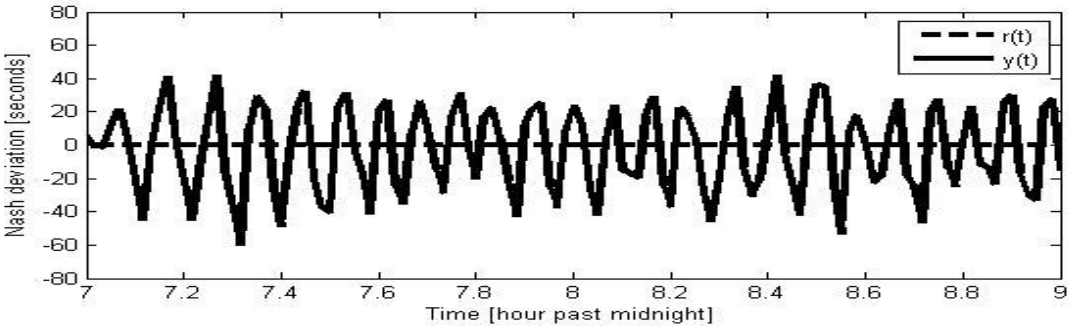

Figure 31. Output on a normal day with disturbance and reactive bang-bang control.

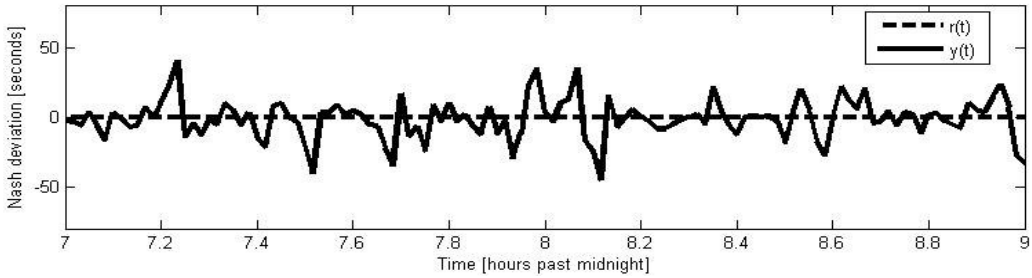

Figure 32. Output on a normal day with disturbance and predictive bang-bang control.

In contrast to bang-bang control, neither P-control nor PI-control makes the system performance significantly worse on a normal day with disturbed output.[106] This is most easily understood if one takes a look at the input signal. In Figure 33, the input from the PI-controller optimized in Chapter 4.1.1.5.1 is depicted (the input from the P-controller is very similar).

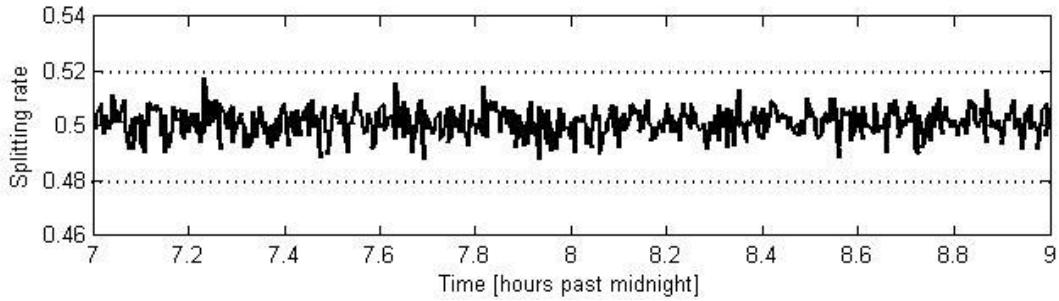

Figure 33. Input from the PI-controller optimized in Chapter 4.1.1.5.1. The area between the two broken lines is the dead-zone.

The most striking feature of this input signal is that it is always inside the dead-zone. That means that no guidance is given, which in this situation implies that the system behaves equally as with no guidance at all (Figure 30). Hence, no significant differences are observed.

A prerequisite for the good results with P-control and PI-control in a disturbed system is that the dead-zone is big enough. If the input is allowed to go outside the dead-zone area, the good behavior cannot be guaranteed. Obviously, the ratio between disturbance variance and dead-zone size is important. When the disturbance is large, the input increases as well and it might end up outside the dead-zone. The choice of the dead-zone should therefore take the impreciseness of the output measurements into consideration. Figure 34 shows the output of the system, when PI-control without dead-zone is applied. The figure clearly depicts what happens if the dead-zone is not big enough. The input-plot is found in Figure 35.

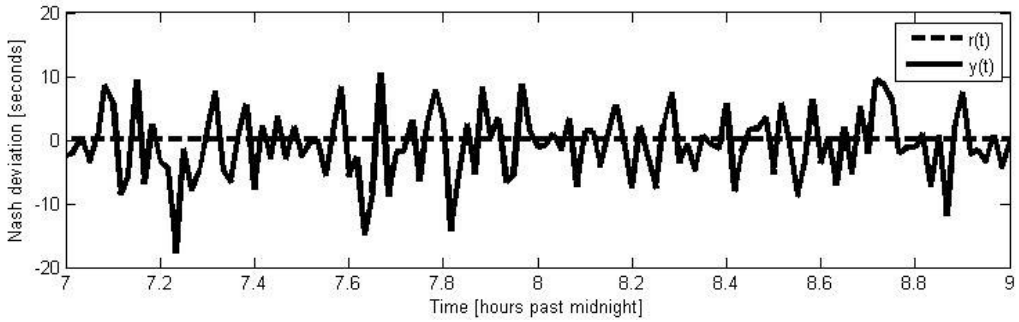

Figure 34. Output from the PI-controller optimized in Chapter 4.1.1.5.1 without dead-zone. Normal day with disturbed output.

[106] The evaluation measures from these experiments are found in Appendix B.

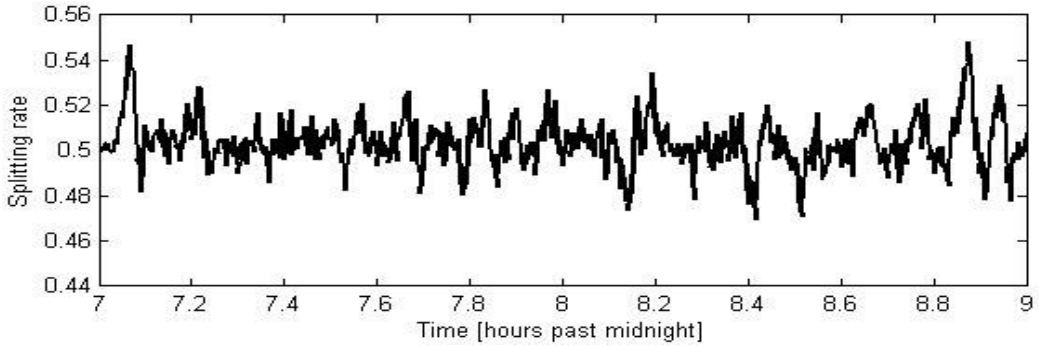

Figure 35. Input from the PI-controller optimized in Chapter 4.1.1.5.1 without dead-zone Normal day with disturbed output.

The plot in Figure 34 shows that the PI-control makes the system output worse than without any control at all (Figure 30) if the controller dos not have a dead-zone. The aggregated values *AN*=13.4, and *AD*=367 are significantly higher than when no control is applied. They are however significantly better that the corresponding values for system controlled with bang-bang controller, also when predictive travel times were used in the control output.[107] The conclusion that can be drawn is that the dead-zone is a very important parameter if one wants the control to work well also at times when no control is needed. When a sufficiently big dead-zone is included in the controller, the controller does not run the risk of distorting a good traffic condition. As soon as the dead-zone is taken away (or is too small), the risks get larger. It is also important to remember that the bang-bang controller can not have a dead-zone since the output is always at one of the boundaries.[108] This explains why this control strategy is so sensitive to disturbances on the output in situations when the traffic conditions are unproblematic.

4.1.3 Robustness and sensitivity testing

In Section 4.1.1 it was shown that feedback control improves the system performance considerably when there is an accident on one of the routes in the small test network. The improvements are dependent on the control strategy used; the biggest improvements are observed when the system is controlled by a well tuned PI-controller, whereas bang-bang control improves the system performance more modestly. A good controller should however not only work well for the situation it is tuned for. Optimally, a feedback controller should be able to handle a large amount of different scenarios, always keeping the output close to its reference value. In Section 4.1.2, the controllers' ability to achieve and keep a Nash deviation was investigated in a normal-day-situation. I.e. the controllers were tested in a situation that they were not mainly constructed to handle. When such a testing is carried out, one is testing a *robustness* aspect of the controllers. In this section, a few other such robustness tests are discussed.

---

[107] The PI-control without dead-zone did not generate any significantly different results if predictive travel times were used instead of the normal reactive.

[108] It would indeed be possible to implement a dead-zone also in a bang-bang controller. A discussion about this topic is found in Chapter 5.3.

The parameters varied in the tests are accident size, accident location and compliance rate. In addition, the system behavior with differently large output disturbances is also tested. This last test says something about the controllers' disturbance sensitivity.

4.1.3.1 Variations in accident size

In order to test the different controllers in situations with differently large accidents, *AN*-values[109] from simulations with different capacity on the link with the accident (link 6, Figure 2) were collected. The capacity, which in the standard "Accident-scenario" was 1300 cars/hour, was set to even multiples of 200 between 200 and 1600 cars/hour. Since the demand is approximately 1500 cars/hour, higher capacities are not interesting to test. The control strategies tested were reactive P- and PI-control. The bang-bang control tested was however predictive, since that was shown to work better than reactive in Section 4.1.1.3.The results are depicted in Figure 36. In Figure 37, the most interesting part of the plot is zoomed.

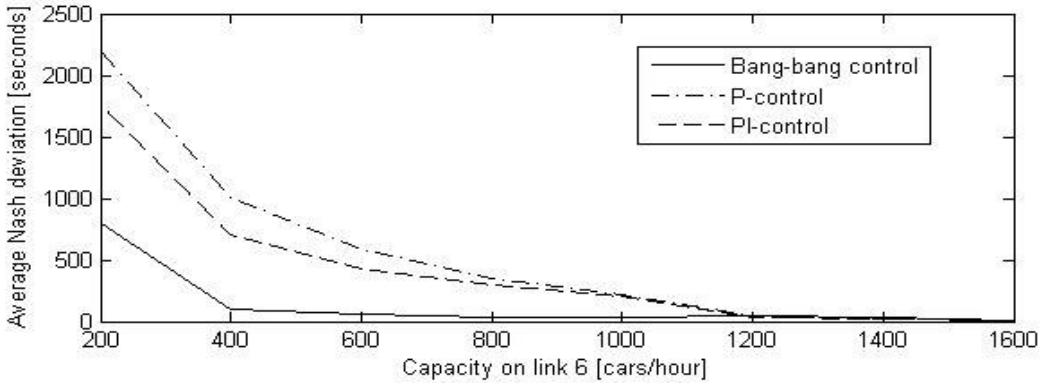

Figure 36. System performance with different controllers and different accident size.

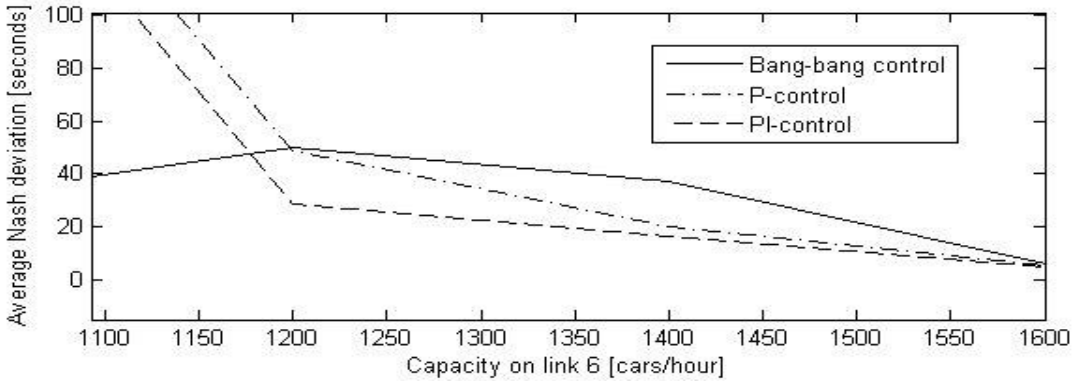

Figure 37. System performance with different controllers and different accident size. A zoom of the most interesting part of Figure 36.

[109] Here, as in all other robustness experiments on the small test network, the *AN*-values are averages from ten simulation runs.

The P-controller and the PI-controller are tuned for a capacity of 1300 cars/hour on link 6. Non-surprisingly, Figure 37 shows that a PI-controller gives the best system performance when the capacity is close to that value. P-control is second best and bang-bang controller is the worst.[110] This is however only the case when the capacity is higher than 1200 cars/hour. Already at 1100 cars/hour, the bang-bang strategy is significantly better than both P-control and PI-control. In Figure 36, one sees that the bang-bang controller works much better than the two other when the capacity is very limited. For example if the capacity is 500, both P-control and PI-control give catastrophic *AN*-values that are higher than 500 s. With bang-bang control, the corresponding value is lower than 60 s. The conclusion one can draw is that bang-bang control is much more robust than P-control and PI-control when it comes to the size of the accident. That means that it is probably smart to control the system by means of a bang-bang controller if one is not sure of what disturbances that will occur in the system. In practice, P-control and PI-control are so bad when the accident is large, that it is sensible to use these strategies only when the capacity reduction is somewhat controllable, e.g. when one or two lanes are blocked due to road constructions.

One other way of looking at the robustness against accident size is presented in Figure 38. Here, the relative improvement of *AN* for different controllers in relation to no control at all, is plotted. One can clearly see that whereas the bang-bang control improves the system considerably for almost all system settings, P- and PI-control control well when the capacity is close to 1300 cars/hour, but rather poorly for other capacities.

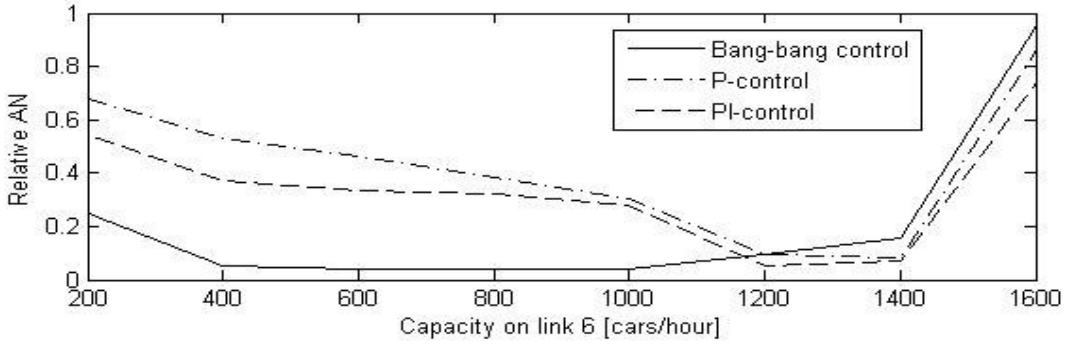

Figure 38. Relative improvement in comparison with no guidance, when different control strategies are applied and the capacity reduction is varied.

4.1.3.2 Variations in accident location

Another way of evaluating the robustness of the controllers is achieved by varying the location of the accident. Practically, that meant that the length of link 5, immediately prior to the accident on link 6, was varied. The link length, that normally is 2000 meters, was set to values between 500 meters and 9500 meters. Remembering that predictive travel times should have the ability of handling time lag problems better than

[110] As was shown in Section 4.1.1 these differences are significant.

reactive (Section 2.2.5), simulations were run with predictive as well reactive travel times in the control output for all three control strategies. For the P- and PI-controllers, the parameters found optimal in Section 4.1.1 were used.

For PI-control, no significant difference was experienced between the reactive and the predictive control. For both bang-bang control and P-control, the predictive mode was significantly better. As is shown in Figure 39, the difference with P-control grows bigger when the distance to the accident increases. The difference becomes significant when the distance is 3500 meters or longer.

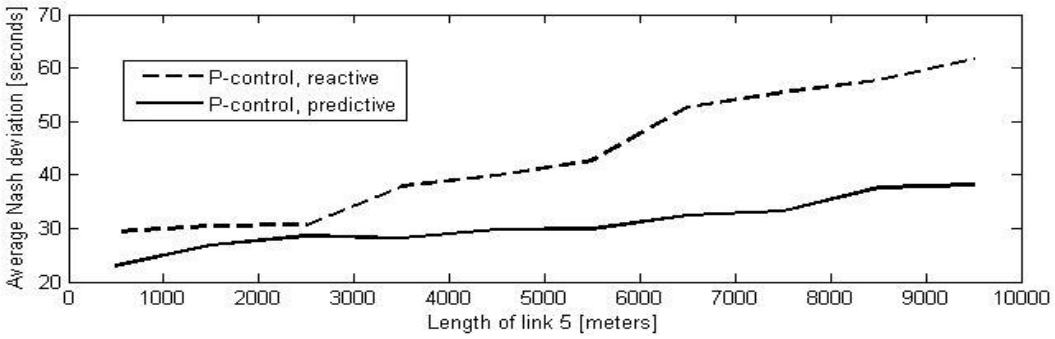

Figure 39. AN-values for simulations with different P-controllers when the accident is located differently.

The difference between reactive and predictive measurements is significant for the P-controller. The difference is however rather modest in contrast to the differences observed when the bang-bang controller was tested in the same way. Figure 40 shows the same test for the bang-bang controller.

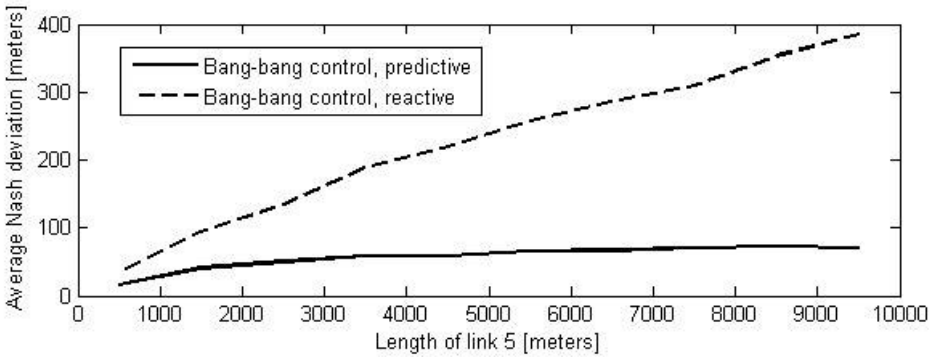

Figure 40. AN-values for simulations with different bang-bang controllers when the accident is located differently.

For the reactive bang-bang controller, the system output is not at all acceptable when the accident is located far away from the guidance spot. In fact, when link 5 is 9500 meters long, this controller does only improve the system output marginally by approximately 15 %. In contrast, the predictive bang-bang controller handles the long time lags much better.

When all three control strategies were to be compared, predictive controllers, being better or equally good, were chosen for all three strategies. The results are shown in Figure 41.

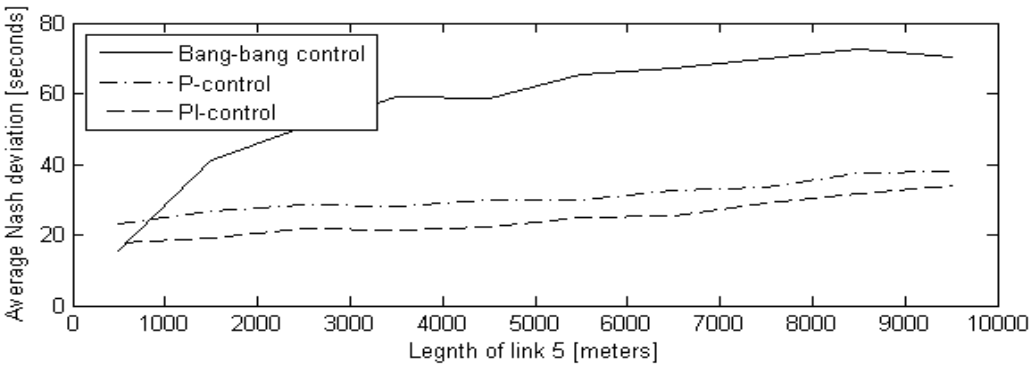

Figure 41. Comparison of all three control strategies in predictive mode.

Figure 41 shows that both P-control and PI-control are rather insensitive to location variations of the accident. The PI-controller is always significantly better than the P-controller. The graph of the bang-bang controller is more interesting. Also with a predictive control output, the location of the accident has a big impact on the system performance when this control strategy is applied. When the accident is located far away, the bang-bang controller generates results that are much worse than the *AN*-values achieved with P- and PI-control. The results are up to three times as bad. With small time lags, this difference is decreased however and for the very shortest link length tested (500 meters), the bang-bang controller outperformed the more sophisticated control strategies.[111]

The conclusion one can draw from this way of evaluating the robustness is that predictive measurements become more and more important as the time lags grow bigger. Moreover, one can conclude that neither the P- nor the PI-controller is particularly sensitive to the location of the accident. In contrast, the bang-bang controller is highly sensitive to such variations. Bang-bang control should therefore only be used in situations when the time lags are short. When the accident is far away from the point of guidance, bang-bang control is better avoided.

4.1.3.3 Variations in compliance rate

A parameter that is hard to estimate correctly is the compliance rate. How big a fraction of the population that actually follows the guidance is likely to vary depending on factors such as where in the city the guidance is applied and what people the guidance is given to. Compliance rate robustness is therefore a very important feature for every controller. Simulating in the “Accident scenario” with different compliance rates and

[111] It is important to remember that the P- and PI-controllers tested were tuned for a length of link 5 of 2000 meters. It is likely that both a P- and a PI-controller, optimized for a shorter link length would still be superior to the bang-bang controller.

the controllers from Section 4.1.1 the following results were given. The bang-bang controller used predictive travel times, whereas the two other controllers were based on reactive travel times.

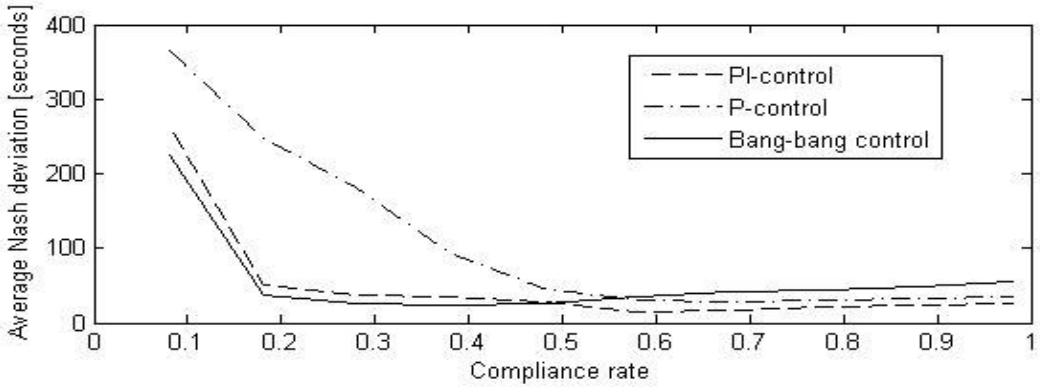

Figure 42. AN-values when the compliance rate is varied.

Figure 42 shows the results of the simulations with different compliance rates. When the compliance rate has its default value, 0.8, the results from Section 4.1.1 are achieved; i.e. the PI-controller is the best, the P-controller is the second best and the bang-bang controller gives the worst system performance. When the compliance rate is really low, the system behaves as if there was not any control at all and *AN* thus tends towards 400 s (as in Section 4.1.1.1). Interestingly, when the compliance rate is in the interval $[0.2,0.5]$, the P-controller is much worse than the two other control strategies. In this interval, the bang-bang strategy is significantly better than the other two, although PI-control is only slightly worse.

The reason for the big control performance differences when the compliance rate is low is not totally obvious. It is however fully possible to understand the robustness differences, if the ways the three control strategies actually control are considered. The good robustness of the bang-bang controller is most intuitive. This controller typically controls the input too hardly and it is thus rather an advantage if fewer agents comply with the guidance given, since this means that the control is somewhat softened. Looking at Figure 42, it is clear that the lower compliance does improve the bang-bang controller, which is optimal for `compliance=0.4`.

In contrast to the bang-bang controller, the P-controller does not control too hardly when the compliance rate is set to its default value 0.8. Contrary, the P-controller is optimized for this compliance rate value and the control is thus neither too hard nor too soft. When the compliance rate decreases, the control is not strong enough. The result of this too soft control is worsening system performance.

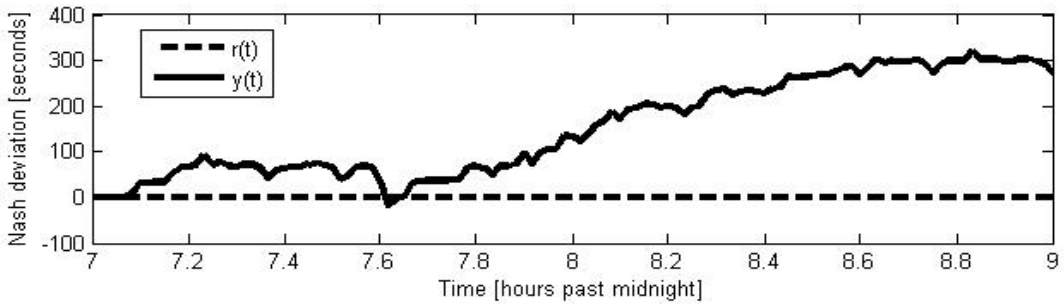

Figure 43. Output with P-controller. The compliance rate is 0.3.

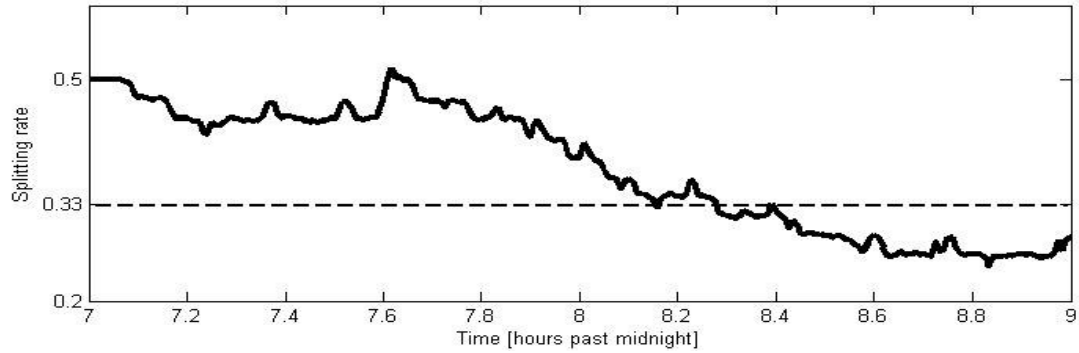

Figure 44. Input with P-controller. The compliance rate is 0.3.When the input is below the broken line, the agents are guided along route 2 all the time.

In Figure 43 and 44, the output and input of the system, controlled by the P-controller when the compliance rate is 0.3, are depicted. Obviously, the output deteriorates heavily from the reference value and the control signal is not doing enough to compensate for the exceeding output. Interestingly, after approximately 8:15 AM, $u(t) < 0.33$. That implies, according to the mapping in Section 2.2.6, that the entire traffic is guided along route 2. In other words, the P-controller behaves exactly as a bang-bang controller from that moment on and the controller is doing as much as it possibly can to decrease the output. It is however not enough to guide all travelers along route 2 at this point in time, since the output continues to increase all the way up until 9:00 AM, although at a slightly lower rate.

In order to understand why the PI-controller does not have the same problems as the P-controller when the compliance rate is low, a comparison between the output and the input for the two controllers is needed. In Figure 45, the output from the system, controlled by the PI-controller when the compliance rate is 0.3, is depicted.

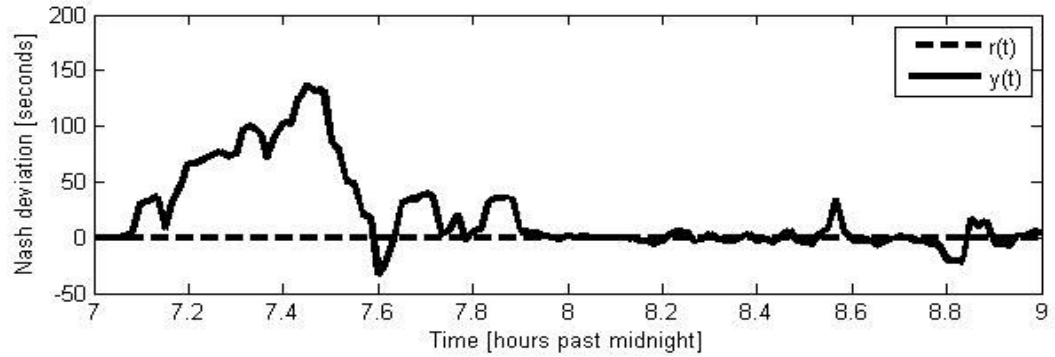

Figure 45. Output with PI-controller. The compliance rate is 0.3.

Also when the PI-controller is applied, the output initially rises quickly. In contrast to the output from the system controlled by a P-controller, the rise stops abruptly around 7:30 in AM, and from that time on, the system output is kept close to the reference signal during the entire simulation. The input signal, depicted in Figure 46, explains why this happen.

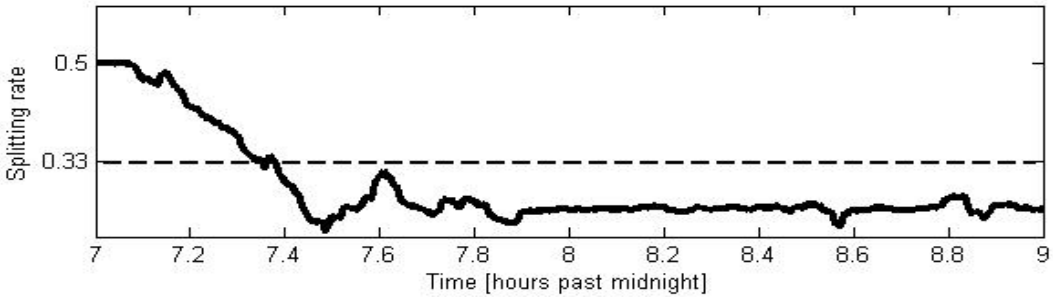

Figure 46. Input with PI-controller. The compliance rate is 0.3. When the input is below the broken line, the agents are guided along route 2 all the time.

What is striking with this input signal in comparison with the input from the P-controller in Figure 44 is that it reaches the critical level of 0.33 much quicker. The reason for that is that the integral in the controller quickly increases when the output starts to rise as in Figure 45. Already from approximately 7:30 AM, all agents are guided along route 2 when PI-control is used. Contrary to what happens when the input from the P-controller reaches 0.33, this maximal guidance is successful in this case. I.e. the output in Figure 45 immediately goes down towards zero, whereas in the P-controlled system (Figure 43) the effect is not very big at all. This might seem intuitively non-logical; the same guidance gives apparently totally different effects in the same traffic system. The difference is however that in the PI-controlled case, the maximal guidance was applied already at $y(t) \approx 130$ whereas when the P-controller started to guide all agents into route 2, $y(t)$ was already larger than 230 seconds. Such a difference might have a substantial impact in a dynamical system and especially in a system like this that is both non-symmetric and highly non-linear.

The differences in compliance robustness between P- and PI-control can however also be explained in a more mathematical way. When the compliance rate is decreased, the effect of static inputs on the system output is obviously also decreased. Another way of expressing that is that the *static gain* of the system decreases with decreasing compliance rate. The static gain is usually denoted $G(0)$. Moreover, the static gain is indirectly determining the magnitude of the static error of the controlled system. For a system controlled by P- or PI-control, the static error e_{static} is determined by the following equation.[112]

$$e_{static} = \frac{1}{1 + G_c(0) \cdot G(0)} \qquad (16)$$

In this formula, $G_c(0)$ is the static gain of the controller. For a P-controller, $G_c(0) = K_P$. This means the denominator in (16) decreases when the compliance rate and $G(0)$ gets smaller. Hence it follows that the static error grows larger. [113]

In contrast to the P-controller, $G_c(0) = \infty$ for a PI-controller. The denominator in (16) is therefore infinitely large as long as $G(0) \neq 0$, which is true for all system where the guidance actually has an impact. The static error is therefore always zero for a PI-controller, which is obvious in Figure 45.[114] The robustness against decreasing compliance rate is thus much better when an integral part is added to the proportional controller.

The conclusions one can draw from this experiment is that P-control is not very good to use in situation where the compliance rate is uncertain. PI-control is much better than P-control at handling compliance rate changes. Bang-bang control works however well in almost all cases and when the compliance rate is low, this strategy is actually the best of the three.

4.1.3.4 Control with disturbance

Finally, the system performance for the three control strategies was also tested when a normally distributed disturbance of varying size was added to the output. The simulations were still run in the accident scenario. As before, the bang-bang controller worked with predictive travel times, whereas the other two controllers had the deviation of reactive travel times as their control output. The standard deviation of the disturbance was varied between zero and 20 seconds. The results are depicted in Figure 47.

---

[112] Glad & Ljung (1989), p 53.
[113] ibid.
[114] ibid.

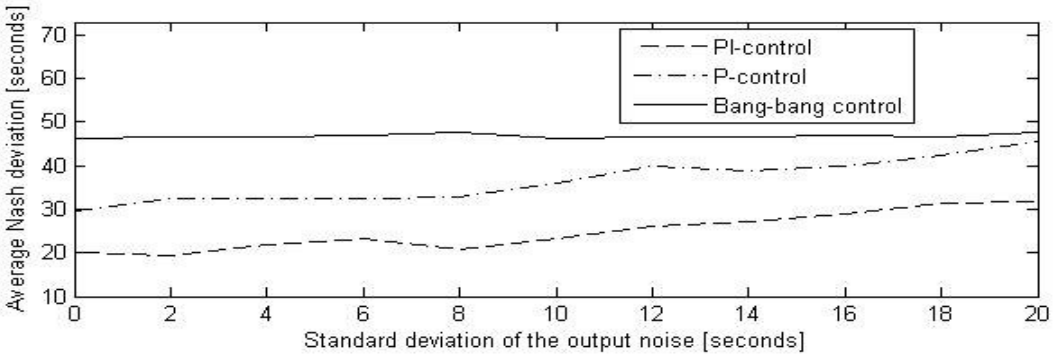

Figure 47. AN-values when the magnitude of the output noise is varied.

Apparently, bang-bang control is totally insensitive to disturbances. The system performances of the P-controller and the PI-controller get significantly worse when the disturbance is increased. It therefore seems reasonable to conclude that it is less important to use sophisticated control strategies such as P and PI-control when the system output is noisy. Nevertheless, remembering the conclusions from the experiments with disturbed output on a normal day (Section 4.1.2.2), one should not use bang-bang control unreservedly when the disturbance is large. The bang-bang strategy might be good at handling the accident situation when the disturbance is big, but as soon as there is no accident in the system; a bang-bang controller, having no dead-zone, has a tendency of worsening the system performance that would have been almost perfect if no guidance at all was applied.

4.1.4 The small test network – concluding remarks

The experiments on the small test network illuminate many aspects regarding automatic feedback control in traffic networks. The simulations discussed in Section 4.1.1 and 4.1.2 shows how different controllers can be tuned and how well different control strategies work in different scenarios. In the Section 4.1.3, the results of different robustness and sensitivity tests have been presented and discussed. These experiments provide insight to the way different controllers work under non-optimal conditions.

The observations from the experiments have made it possible to formulate a few rules of thumb that should guide the control design. The most important such guideline is that a controller works much better in the system it is optimized for, than in systems where some properties are altered. It is thus crucial to tune the controller for the scenario that is most likely to happen. If the problematic scenario can be predicted correctly, an optimized P- or PI-controller will most definitely produce good system performances.

The controllers should always be tuned for the most likely scenario but unfortunately, all scenarios can not be predicted perfectly. It is however very helpful if one can estimate what aspects of this most likely scenario that are the most uncertain. Is the most uncertain simulation parameter the accident size, the accident location, the compliance rate or the disturbance? Depending on the answer to those questions, the control design should be carried out in the following way

- ✓ If the size of the accident is the most uncertain parameter, bang-bang control should be applied.
- ✓ If the location of the accident is most uncertain, PI-control should be applied.
- ✓ If the accident is located far away, bang-bang control should be avoided.
- ✓ If the compliance rate is uncertain, P-control should be avoided.
- ✓ If the disturbance is large, the dead-zone should be chosen large as well. Thus, bang-bang control, having no dead-zone, should be avoided.
- ✓ If bang-bang control is applied, the control output should be based on predictive travel times.

In the next Chapter, these guidelines are used when control is applied in the reduced Berlin network. As will be shown in Section 4.2.3, some of these conclusions are valid also in this larger environment. In the same section, it will however also be shown that a few of the guidelines are not possible to generalize outside the small test network framework.

4.2 The reduced Berlin network

4.2.1 Normal day

In the normal day scenario, with no guidance, the output $y(t)$ from the reduced Berlin network typically looks like in Figure 48.

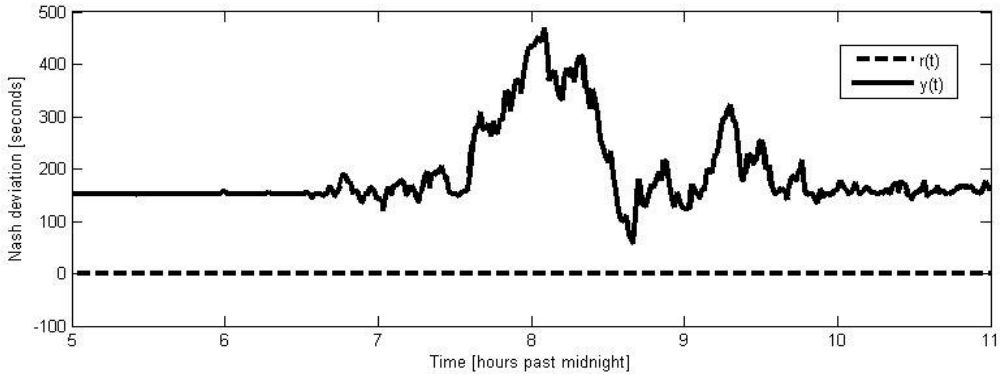

Figure 48. Output from the simulation in the reduced Berlin network. No accident and no guidance.

Looking at this plot, one might conclude that the system is not at all in Nash equilibrium. The travel time on route 1 (i.e. the small inner city route in Figure 3, section 2.3.2) is always larger than the travel time on route 2 (i.e. the highway). Hence, $y(t) > r(t) = 0$ always holds. The plot of the disbenefit-value $d(t)$ in Figure 49, does however tell a different story.

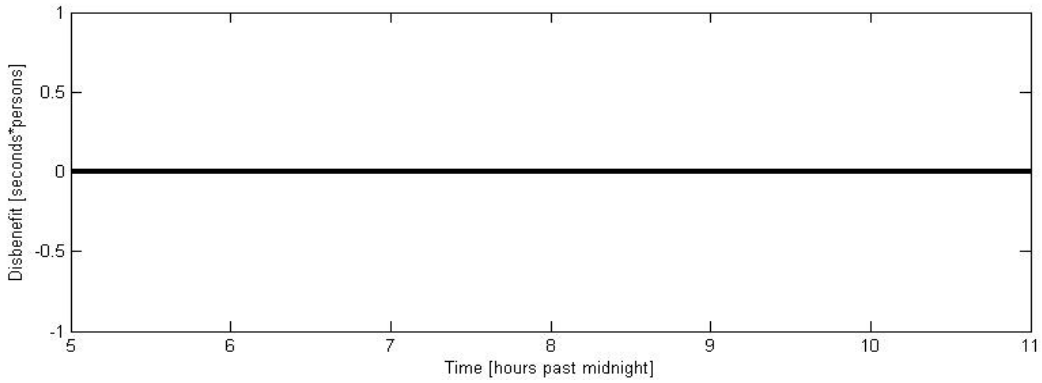

Figure 49. Disbenefit plot with neither accident nor guidance.

Apparently, the entire traffic load between the sign-links and the destination-links uses the faster highway. The system is thus in perfect Nash equilibrium, even though the system output is constantly larger than the reference value. The aggregated evaluation measures confirm this conclusion. *AN*=224 and *NM*=204 indicate that the system is not in an equilibrium. *AD*=0 does however confirm the conclusion from Figure 49 that a Nash equilibrium is indeed achieved.

In contrast to the small test network simulations, there is no clear resemblance neither between $y(t)$ and $d(t)$ nor between the corresponding *AN*- and *AD*-values when the reduced Berlin network is simulated. Seeing that $y(t) = 152$ in a free flow situation (the first two hours in Figure 48), *AN* does not give a very good indication of the system performance. In the following, *AD* has therefore been the evaluation measure considered most relevant to study. It is also this quantity that has been used most frequently when the different control strategies' abilities to maintain Nash equilibrium have been compared and evaluated. All evaluation measures from all simulations are found in Appendix C.

In Section 4.1.2.2, the controllers' ability to handle a normal day scenario with disturbed output was tested. This kind of investigation is not relevant to carry out in this larger case. The reason for this is that, provided that `nominalSplitting=0`, no controller guides any car into route 1 as long as $y(t) > 0$. As one can see in Figure 48, that is always true in a normal day scenario. Moreover, one would need to add an unreasonably large disturbance to the output to change that.[115] Hence, no such simulations have been run in the larger network.

4.2.2 Accident case

As is thoroughly described in Section 2.3.2, a capacity reduction representing a traffic accident on link 7832 was introduced in this scenario. The accident was then being controlled by means of the different control strategies discussed in Chapter 2.2.

[115] When the output is at its lowest point around 8:40 AM, an output disturbance of approximately 50 seconds would have been needed to push the output below zero.

4.2.2.1 No guidance

When the accident scenario is simulated without guidance the output typically looks as in Figure 50.

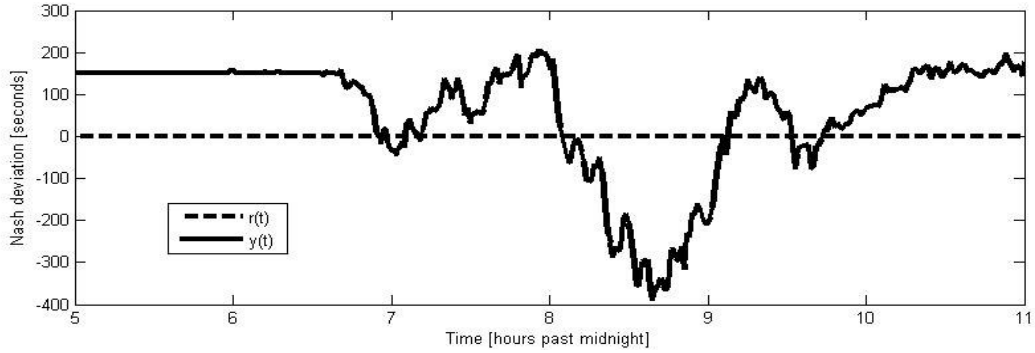

Figure 50. Output with accident but without guidance.

In comparison with Figure 48, one can clearly see that the accidents affect the system output. Before 6:30 AM, the traffic load is so small that route 2 can handle the entire demand, also with the capacity reduction on link 7832. From that time on, congestions start to build up on route 2, increasing $TT_R^2(t)$ and decreasing $\Delta\tau_R(t) = y(t)$. From approximately 8:00 AM up until 9:00 AM, route 2 is so congested that $y(t) < 0$ for a long period of time. The plot in Figure 50 is clearly different than the one in Figure 48, but the plots alone do not show that the system performance is worse with than without accident. *AN*=151 and *NM*=51.6 rather indicate that the system is closer to Nash equilibrium when there is an accident on route 2. Once again, $d(t)$ in Figure 51, gives a better description of the situation.

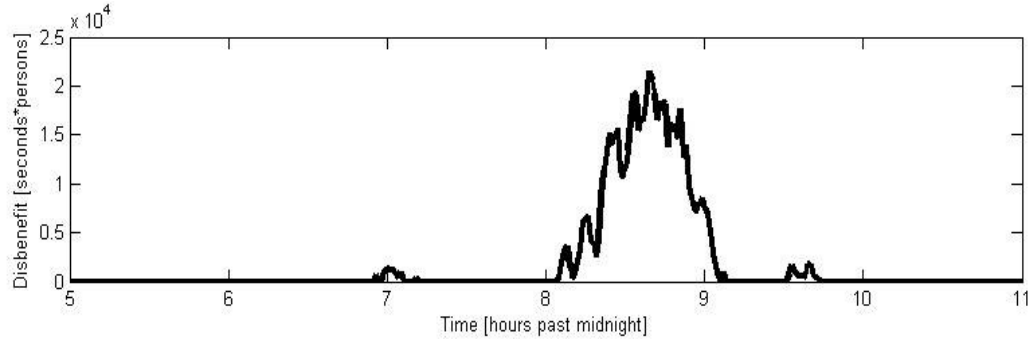

Figure 51. Disbenefit valued, d(t), with accident when no guidance is applied.

Apparently, the accident does not make the system performance better. In contrast, since all travelers use route 2 many people do actually travel on the slower route, when the highway is the slower of the two routes between 8:00 and 9:00 AM. In this interval, the plot in Figure 51 also depicts positive disbenefit-values. The average disbenefit-value (*AD*) is 1959 seconds·persons. One can thus conclude that the accident does lead to a traffic situation that is not in a Nash equilibrium. Hence, guidance has a potential of improving the situation.

4.2.2.2 Static guidance around the accident

Also for the simulations in the reduced Berlin network, static guidance around the accident was tested. The output from this simulation is shown in Figure 52 and the disbenefit-plot is depicted in Figure 53.

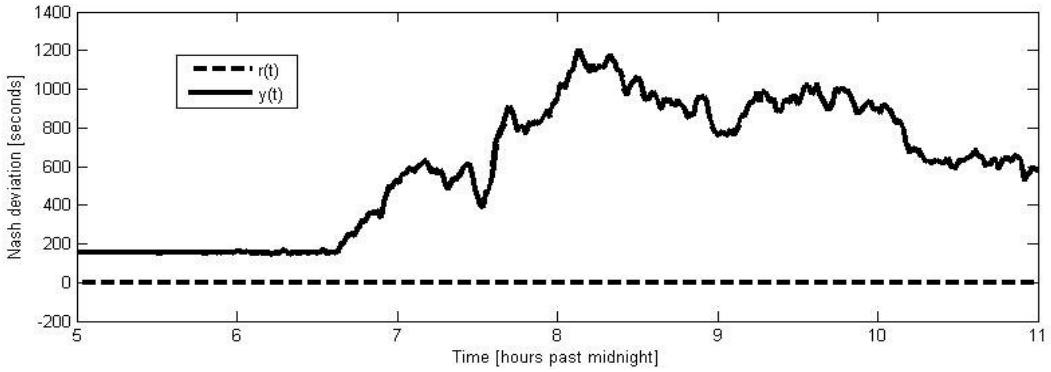

Figure 52. Output when all travelers are guided around the accident.

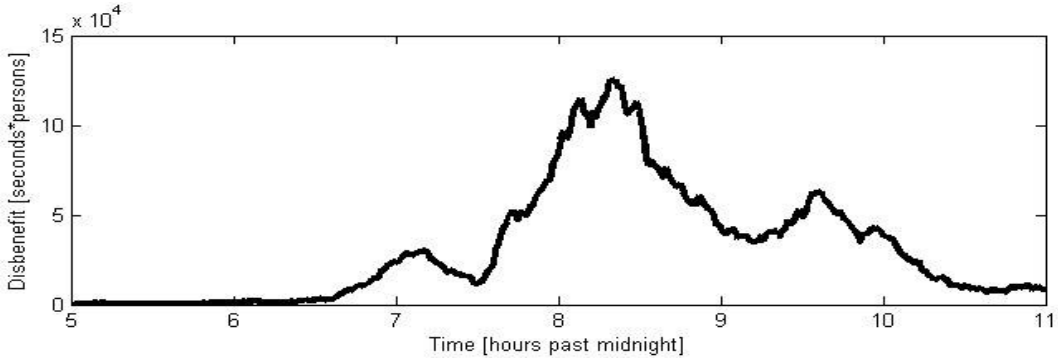

Figure 53. Disbenefit-values when all travelers are guided around the accident.

The figures show that static guidance is not a good idea in this situation. Route 1, being an inner-city road, has much less capacity than the freeway (route 2). This also holds when an accident is generated on the highway. Guiding all agents into route 1 therefore leads to even more congestion than if no guidance is applied. Hence, $y(t)$ grows big and since all travelers are guided into this road,[116] $d(t)$ also increases. In comparison with the no-guidance simulation, *AD* is almost 16 times as bad when the static guidance is applied. The conclusion one could draw from this experiment is thus that static guidance should absolutely not be applied in this situation.

4.2.2.3 Bang-bang control

Seeing in the preceding section that static guidance does not solve the problem caused by an accident on route 2, a dynamic bang-bang control was tested. First bang-bang

[116] The compliance rate is still 0.8. Therefore, only 80 % of the agents are actually travelling on route 1.

control with reactive travel-times as control output was applied. Figure 54 shows the output and Figure 55 shows the input from this experiment.

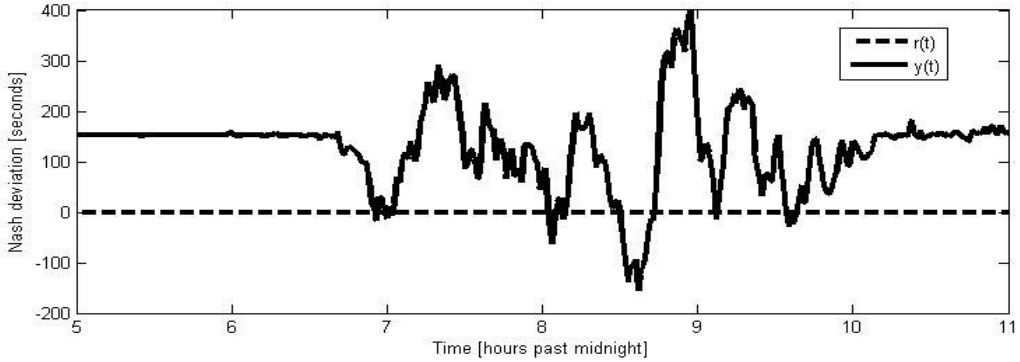

Figure 54. Output when reactive bang-bang control is applied.

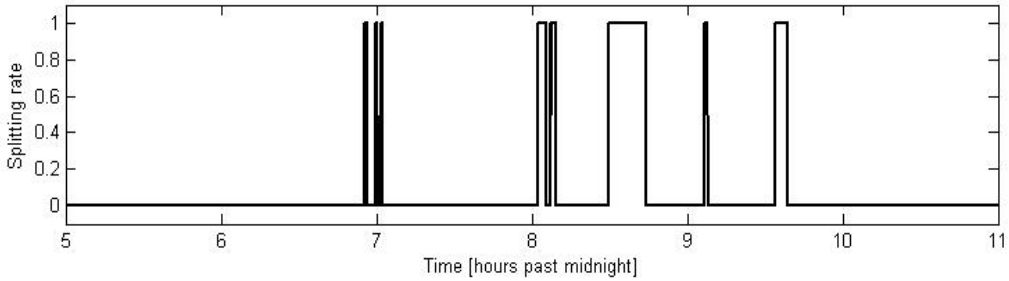

Figure 55. Input when reactive bang-bang control is applied.

In these figures, one can see how the bang-bang controller sends all traffic into route 1 as soon as the output goes below zero. The output is in this way kept above zero most of the simulation time. The oscillating system behavior, typical for bang-bang controlled systems is not so obvious in the plot. Nevertheless, the output does not say anything about how well the Nash equilibrium is maintained. This is more easily seen if one studies the disbenefit-value plot depicted in Figure 56.

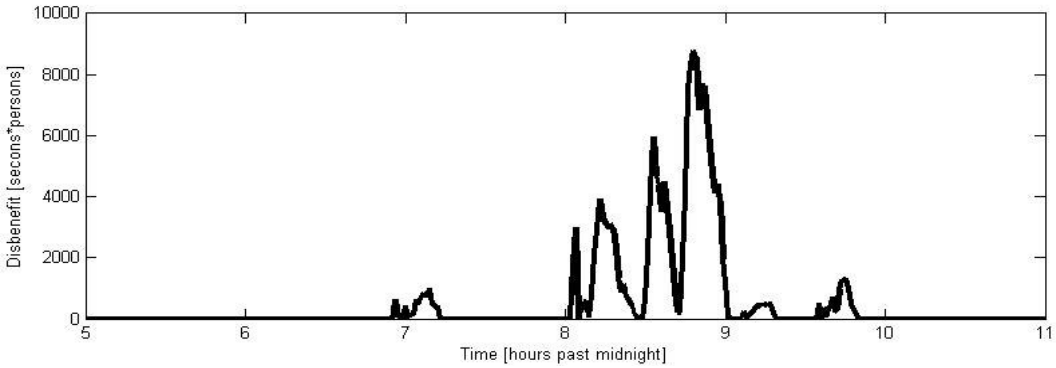

Figure 56. Disbenefit-values when reactive bang-bang control is applied.

If the disbenefit-plot in figure 56 is compared with the corresponding plot without guidance (Figure 51), one can clearly see that bang-bang control improves the system performance considerably. The average disbenefit (*AD*) is now 274 seconds·persons, which is only 14 % as much as the corresponding value without guidance. This difference is of course significant.

The reactive bang-bang controller does improve the system behavior considerably. Remembering from Section 4.1.1.3 that the bang-bang controller worked even better when predictive travel times were used, it seems reasonable to assume that the improvement could be even bigger if the control output were predictive. The disbenefit-plot in Figure 57 from the simulation with predictive bang-bang control does however show this assumption to be invalid. Contrary, the disbenefit becomes slightly larger when the control is predictive. *AD*=313 confirms that the assumed improvement was not materialized.[117]

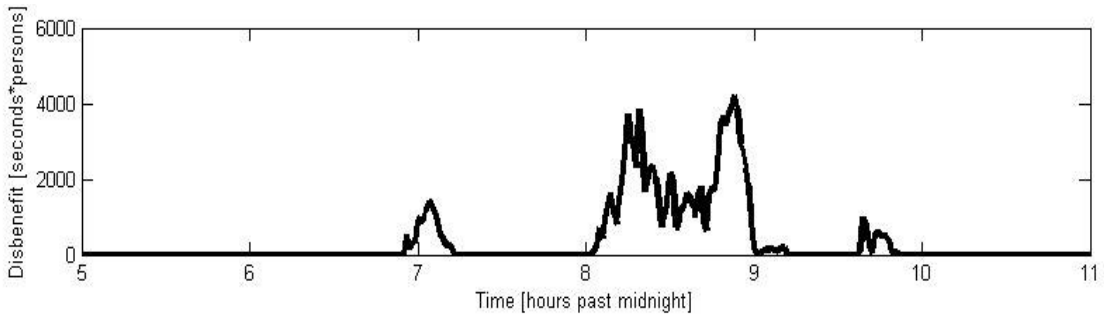

Figure 57. Disbenefit-plot from the simulation controlled by a predictive bang-bang controller.

The results become even more surprising when the lengths of the routes are taken into consideration. The routes and the time lags are actually slightly longer in the large scale scenario and it should thus be even more important to use predictive travel times when bang-bang control is applied.[118]

Presumably, the reason for why predictive travel times, that were shown to be crucial for good bang-bang control in the small test network, do not change the performance significantly in the large network, has to do with the way the predictive travel times are measured. As was noticed in equation (6) section 2.2.5, predictive route travel times are a sum of all predictive link travel times. The prediction is supposed to predict the travel time of the agent guided at time *t*. This way of predicting the travel times does however only generate really good predictions for the first link on the route.[119] In the small test network, the accident was situated on the second link of route 1. The congestion was therefore located on the first link on this route. Hence, the travel time on this first link was the most relevant one when the route travel time was to be measured, since on all

[117] This *AD*-value is however not significantly *higher* than the *AD*=274 achieved by reactive control either.
[118] Discussions about this topic are found in Section 2.2.5 and 4.1.3.2.
[119] s. 16.

other links, the travel times always equaled the free flow travel times. Therefore, the output based on predictive travel times worked really well in this small setting, i.e. they were truly predictive.

In the larger network, the accident is located on the eleventh link on route 2. The predictive travel times are thus not as informative as in the small network. Most definitely, this explains why the predictive travel times do not improve the bang-bang control in the same way in the reduced Berlin network as they did in the small test network setting.

4.2.2.4 Proportional control

4.2.2.4.1 Parameter tuning

As already mentioned, separate tuning of all relevant parameters was not possible in the larger simulation setting due to the long simulation times. The setting of the `VDSSign`-parameters `deadZone=0.02` and `controlEvents=3` used in the small test network were thus used also in the experiments on the larger network.[120] Seeing that all traffic uses route 2 without guidance, the parameter `nominalSplitting` (u_0) was set to 0.

It has already been noticed several times in Chapter 4.2 that the system performance in the reduced Berlin network is much better described by $d(t)$ and *AD* than by $y(t)$ and *AN*. The tuning of the P-controller for the larger network K_P therefore aimed at finding the controller that minimized *AD* instead of *AN*. Figure 58 shows the result of the tuning experiments. As before, the results are averages from ten simulations for each K_P-value and the tuning was done with reactive travel time measurements.

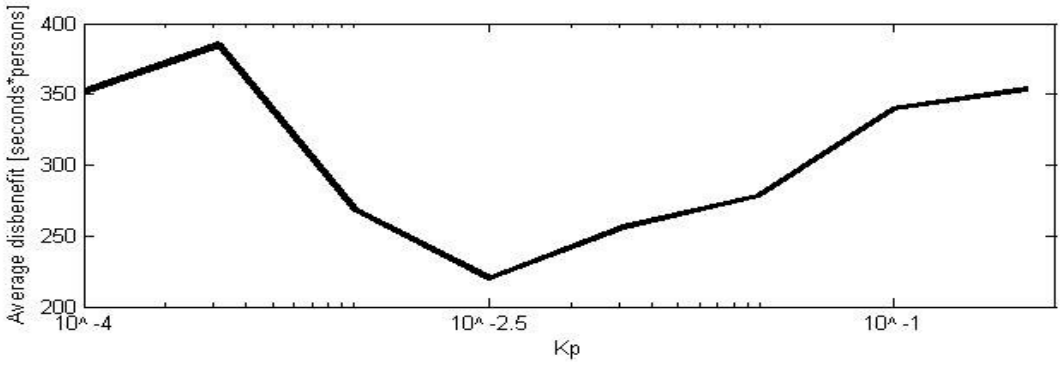

Figure 58. Tuning plot for the P-controller.

The plot clearly shows that $K_P = 10^{-2.5}$ produces the best *AN*-values. This result is significantly lower than for all other K_P-values tested. $K_P = 10^{-2.5}$ was therefore used in the P-controller used to control the traffic in the simulation on the larger network.

---

[120] Admittedly, it is likely that the system performance would have been improved if these parameters had been re-tuned for the larger network. The time available for this master thesis project was however not long enough to tune all parameters in an optimal way.

4.2.2.4.2 Simulation runs

The output signal from the system when it is controlled by the P-controller optimized in the preceding section typically looks like in Figure 59. The more interesting disbenefit plot is found in Figure 60.

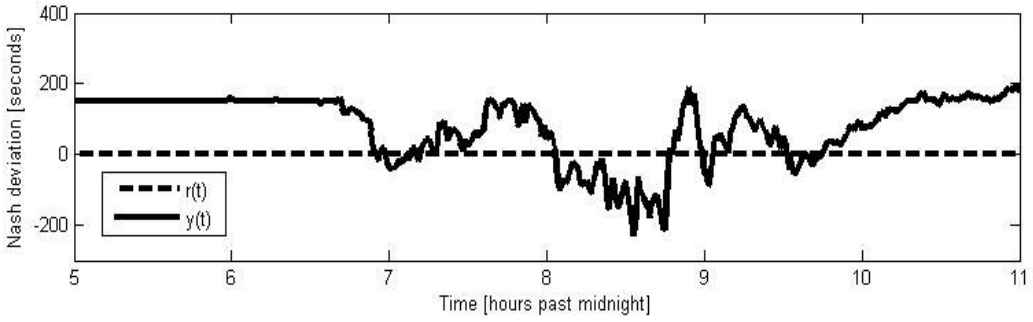

Figure 59. Output with P-control and reactive travel times measurements.

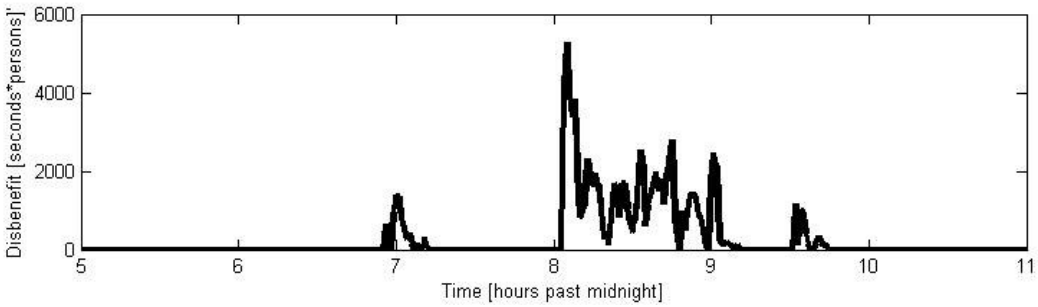

Figure 60. Disbenefit-plot with P-control and reactive travel times.

Comparing the $d(t)$-plot in Figure 60 with the corresponding plot when bang-bang control was used (Figure 54), it is hard to tell which controller that improves the system performance the most. On the one hand, $d(t)$ has a higher maximum when bang-bang is applied, but on the other hand $d(t) \neq 0$ for longer periods of time with P-control. The aggregated quantity *AD* shows that the P-controller is slightly better. In this experiment *AD*=228 seconds·persons with P-control, whereas with bang-bang control *AD*=274 seconds·persons. This improvement is however not significant.

Hence, it does not seem to matter if bang-bang control or proportional control is used to control the traffic on the two routes in the reduced Berlin network. Somewhat surprisingly, a significant improvement was however observed when the P-controller was used with predictive measurements in the control output. Looking at the output and the disbenefit-plot from this experiment (Figure 61 and 62), one sees that this improvement is due to the fact that $y(t) < 0$ for much shorter periods of time. Since a lot more traffic is travelling on route 2, this means that the $d(t)$ is generally lower in this simulation, although $y(t)$ takes large positive values for longer periods of time. The average disbenefit value therefore becomes almost 40% lower (*AD*=143).

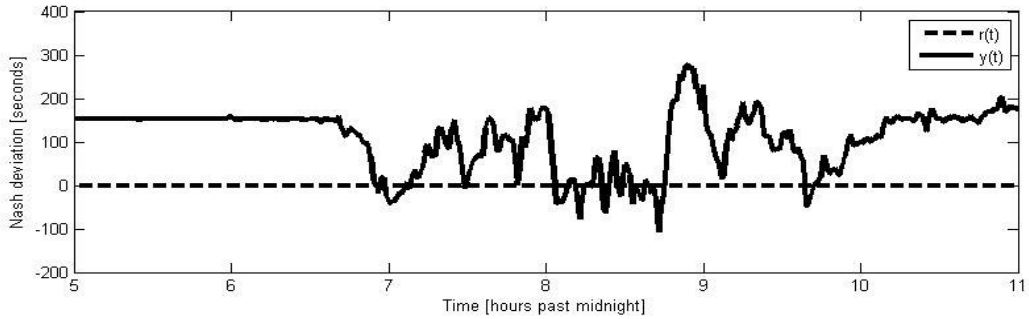

Figure 61. Output with P-control and predictive travel times measurements.

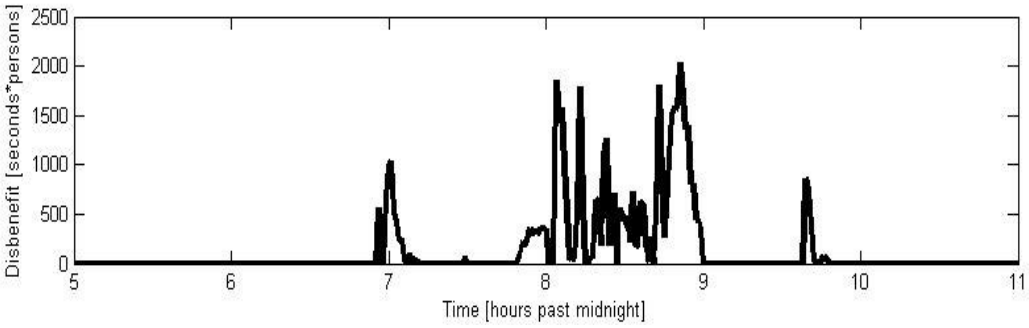

Figure 62. Disbenefit-plot with P-control and predictive travel times.

The conclusions to draw from the experiments with proportional control are firstly that it works well. The most important aggregated measurement *AD* is only 7.2 % as large as it was when no control was applied. Secondly, one can also conclude that the experiments show that no generalizations can be made regarding the effect of the predictive travel times. When the small network was simulated, the predictive measurements improved the bang-bang control but not the P-control significantly. In the larger network, it turned out to be the other way around; the bang-bang control was not improved but the P-control was. The predictive measurements should therefore be considered as a control parameter and it should always be tested by means of trial-and-error.

4.2.2.5 Proportional and integral control

4.2.2.5.1 Parameter tuning

Also for the PI-control, the `VDSSign`-parameters `deadZone` and `controlEvents` were not re-tuned. Thus the setting `deadZone=0.02` and `controlEvents=3` was used. The parameter `nominalSplitting` was set to 0. In the tuning of the control parameters K_P and T_i the Average disbenefit value, *AD*, was used for evaluation. The measurements used in the control output were predictive, since such measurements were shown to work better or equally good for the control strategies already applied to the larger network.

Similarly to when the PI-controller for the small test network was tuned, the K_P-values considered were those surrounding the optimal value for the P-controller.[121] The interval of T_i-values was intentionally chosen really large so that no interesting value could be omitted. The result of the tuning is shown in Figure 63.

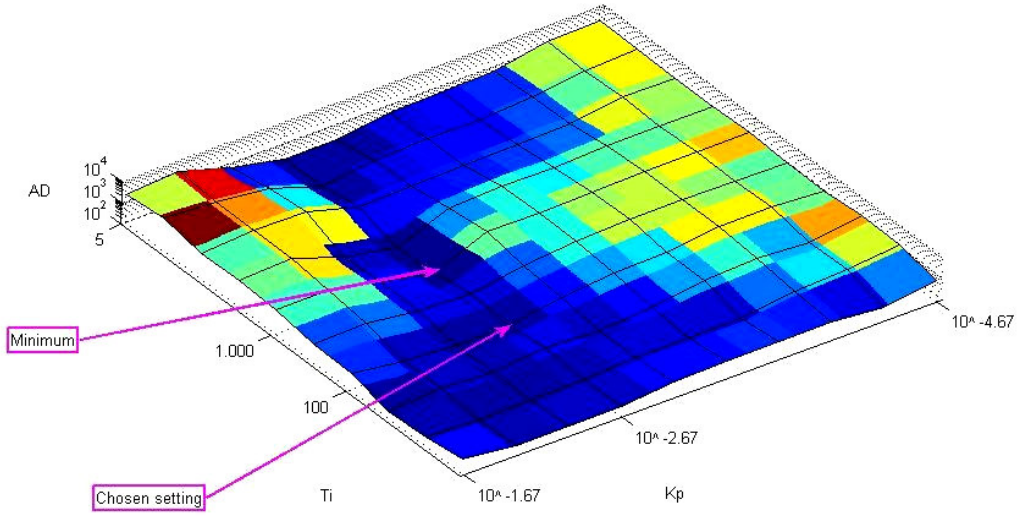

Figure 63. Average disbenefit-value as a function of K_P *and* T_i*. Dark blue color indicates that AD is low. The minimal AD-value is achieved when* $K_p = 10^{-2.6.7}$ *and* $T_i = 10^3$*. The chosen setting was however* $K_p = 10^{-2.6.7}$ *and* $T_i = 10^2$*.*

The parameter combination generating the minimum *AD*-value was $K_P = 10^{-2.6.7}$ and $T_i = 10^3$. At this minimum, *AD*=151 seconds·persons. Remembering from Section 4.2.2.4.2 that the corresponding value with an optimal P-controller was actually slightly lower (143 seconds·persons), one might conclude that there is no point of using PI-control in the larger network. On the other hand, the robustness tests discussed in Section 4.1.3 indicates that controllers which include an integral part are more robust against variations in compliance rate than controllers that are only proportional. It thus seems reasonable to use PI-control also in the larger network, not since it is superior to P-control in the default setting but since it might be better at handling systems where some parameters are altered.

Given the conclusion that PI-control should be applied mainly due to its presumably good robustness, the parameter combination generating the minimum *AD* in Figure 63 does not look very appealing. Changes in simulation parameters such as the compliance rate or the accident size can be interpreted as analogous to small variations in the control parameters. It therefore seems rather risky that the minimum point in Figure 63 is surrounded by much higher *AD*-values. As an example, if K_P is slightly decreased

[121] Following the guidelines from Ziegler-Nichol's method, Glad & Ljung (1989), p 48.

from $10^{-2.67}$ to 10^{-3}, *AD* suddenly skyrockets from 151 to 1572 seconds·persons. Similarly, a small increase in K_P gives an equivalent effect. The parameter setting finally chosen for the PI-controller in the reduced Berlin network was therefore $K_P = 10^{-2.6.7}$ and $T_i = 10^2$. This setting generates a slightly higher average disbenefit-value of 182 seconds·persons. This parameter combination is however located in a much safer area in Figure 63 and it was thus assumed to give the controller better robustness against parameter variations.

4.2.2.5.2 Simulation runs

As expected, the system performance achieved when the system was controlled by the PI-controller tuned in the preceding section was slightly worse than the performance achieved by predictive P-control. In Figure 64 and 65, $y(t)$ and $d(t)$ respectively are depicted.

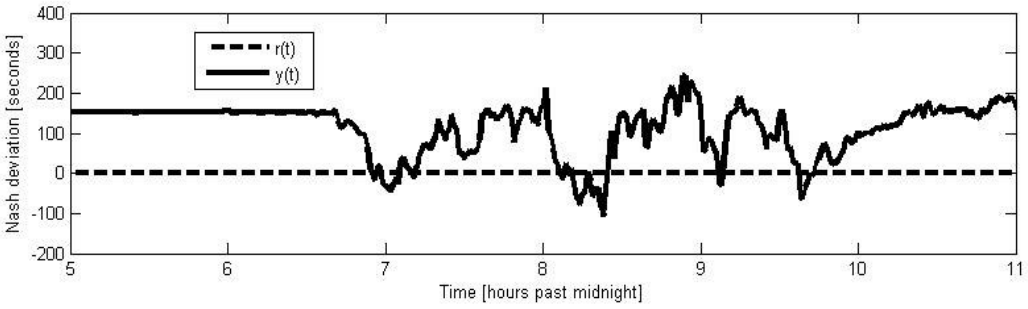

Figure 64. Output when predictive PI-control is applied.

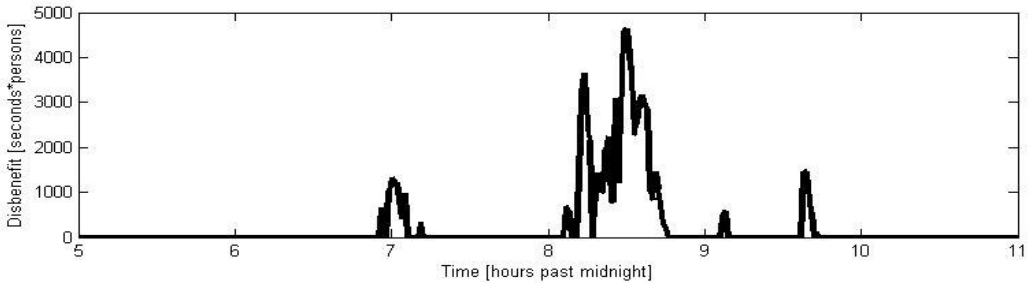

Figure 65. Disbenefit plot when predictive PI-control is applied.

The average disbenefit value *AD* was 231 seconds·persons, which is significantly higher than the corresponding value when P-control was used.[122] The optimal PI-controller is in other words worse than the optimal P-controller at handling the accident in the larger network. It might however have a better robustness against parameter variations than the P-controller. If that is the case will be investigated in the next section of this report

[122] The system was also simulated with a PI-controller that uses reactive travel times in the control output. The results from this simulation show that reactive travel times should not be used in this case. *AD*=515 seconds·persons is by far the highest average disbenefit value achieved by any controller tested.

4.2.3 Robustness and sensitivity

As already mentioned, each simulation takes approximately 60 times as long time in the reduced Berlin network as in the small test network. Systematic robustness testing like the experiments discussed in Section 4.1.3 was therefore not possible to carry out in the larger simulation setting. Instead, only a few tests, intended to check if the conclusions drawn in the small test setting seem to be possible to generalize, were carried out. Moreover, only five simulation runs were performed for each setting instead of the ten runs that have been default in prior sections of the text. Also this reduction in the scope of the experiments was due to the much longer simulation times.

It is important to stress that a few isolated validation tests from one single large network obviously do not make it possible to say that the conclusions from the smaller networks *are* general. Much more systematic experiments from a large amount of networks would have to be made to support such bold claims. In the rest of this chapter, it is thus investigated if the conclusions from the smaller network can be falsified or if it *seems* as if they can be valid generally. It is unfortunately impossible to be more certain that that.

Four different simulation settings were used for robustness testing in the reduced Berlin network: one where the accident size was increased, one where the accident was located further away, one where the accident was really close to the point of guidance and one with lower compliance rate. No experiments with disturbed output signals were carried out. The main reason for that was that the results from the sensitivity experiments in the small test network were not all that spectacular. Validation of the more interesting robustness results was thus prioritized.

For each scenario, the *AD*-values generated by bang-bang control, P-control and PI-control are compared with the corresponding values in the original accident scenario. Predictive travel times, being shown to be better or equally good as reactive travel times for all controllers in the larger network, were used consistently. The *AD*-values from the default setting, used for comparison, are summarized in Table 1.

Bang-bang control	P-control	PI-control
319	143	231

Table 1. AD-values for different control strategies in the default accident scenario.

4.2.3.1 Larger accident

In this setting, the capacity on link 7832 was decreased form 1080 cars/hour to 540 cars/hour. The new capacity represents an accident that is larger than before. The results from the experiment are shown in Table 2.

Bang-bang control	P-control	PI-control
342	230	216

Table 2. AD-values for different control strategies when the size of the accident is increased.

The results from the bang-bang controller and the PI-controller do not change significantly when the accident is increased. The *AD*-value achieved when P-control is applied is however significantly higher than in the default setting. Comparing these results with the results from the small test network (Figure 36, Section 4.1.3.1) some resemblance is observed. Also in the small network, bang-bang control is more robust than P-control when the size of the accident is increased. Looking at Figure 36, one would however also assume the *AD*-value for the PI-controller to increase considerably. Instead, this value decreases when the accident is made bigger. It is possible that this good robustness of the PI-controller is due to the non-optimal but robust PI-tuning that was discussed in Section 4.2.2.5.1.

The conclusion one can surely draw from this experiment is that the robustness results from the experiments with varying accident size in the small test network can only be partly validated. The P-control has problems with the larger accident, whereas the bang-bang controller and the PI-controller handle the changed parameter setting rather well. The difference between the control strategies is however not at all as large as it was when the accident size was increased in the smaller network.

4.2.3.2 Longer time lags

In this setting, the accident was moved from link 7832 to link 7809. The distance from the point of guidance to the accident was by that increased from approximately 3 to approximately 4 kilometers. The results form the simulations are shown in Table 3.

Bang-bang control	P-control	PI-control
143	146	173

Table 3. AD-values for different control strategies when the accident is located further away from the point of guidance.

Remembering the discussion in Section 4.1.3.2, the results in Table 2 are indeed very surprising. The long time lags are, contrary to what was expected, *improving* the system performance and the improvements are largest for the bang-bang control. The conclusion from Section 4.1.3.2 was that long time lags are worsening the controllers' abilities to control and that the problems are most severe when bang-bang control is applied. Obviously, this conclusion was not at all validated by the tests on the larger network.

4.2.3.3 Short time lags

In this setting, the accident was moved to link 8376 approximately one kilometer from the point of guidance. The time lag in the system was thus shortened. The results from the experiments are shown in Table 4

Bang-bang control	P-control	PI-control
290	240	431

Table 4. AD-values for different control strategies when the accident is located closer to the point of guidance.

The results in Table 4 fit better with the conclusions drawn in Section 4.1.3.2 than the results from the experiments with longer time lags discussed in the preceding section. The P-control and the PI-control are significantly worsened in comparison with the default accident scenario whereas the evaluation measure from the system controlled by a bang-bang controller is not changed significantly. That means that the bang-bang controller gets relatively better when the time lags gets shorter. This observation validates the conclusion from Section 4.1.3.2. It therefore seems reasonable to conclude that bang-bang control is more robust against short time lags than the other two control strategies and that this conclusion is valid generally.

The conclusions from the small test network can however not be perfectly validated since shorter time lags in this larger setting implies that the system performance gets *worse* whereas in the small test network it was the other way round.

4.2.3.4 Low compliance

In this setting the compliance rate was reduced from 0.8 to 0.4. Running simulations with different controllers in this setting, the results in Table 5 were achieved.

Bang-bang control	P-control	PI-control
189	474	286

Table 5. AD-values for different control strategies when the compliance rate was decreased to 0.4.

The results fit very well with the results presented in Figure 42, section 4.1.3.3. When the compliance rate is reduced, the bang-bang controlled is improved, the PI-control is slightly worsened and the P-control gets much worse. All these changes are significant. As for the reduction of the compliance rate to 0.4 in the small network, bang-bang control was the significantly best control strategy also in this experiment.

The conclusion drawn in Section 4.1.3.3 that P-control is very bad at handling compliance rates that it is not tuned for thus seems to be possible to generalize. Furthermore, the bang-bang controller's good ability to cope with compliance rate changes seems to be a conclusion that is possible to generalize as well.

4.2.4 The reduced Berlin network – concluding remarks

The most important conclusion one can draw from the experiments carried out on the reduced Berlin network is that guidance really works. Every single controller and parameter settings tested improved the system performances (*AD*) in comparison with the non-controlled case.[123] In comparison with the experiments that were run on the small test network, it is however much more complicated to draw clear conclusions about the different control strategies advantages and disadvantages in this case. Moreover, out of the six rules of thumb proposed in Section 4.1.4 only two seem to be possible to generalize outside the smalls scale setting. These two are firstly that bang-bang control gets relatively better as the time lags get shorter and secondly that the

[123] A summary of all simulation results is found in Appendix C.

robustness against compliance rate reductions is much worse for the P-control than for the other two control strategies.

Apart from these two general conclusions, the guidelines for how to tune design the controller optimally are few and vague. This follows, since it is more or less impossible to know exactly how a variation of a control parameter might affect the system performance. This is for example seen if one observes how differently the use of predictive measurements affects the system behavior in the two networks. The conclusion from the small network[124] was that predictive measurements should be used for bang-bang control but that it does not make a big difference if one uses these or the reactive measurements when P- and PI-control is applied. In the larger network, one could instead observe that predictive travel times are crucial for good P- and PI-control whereas it does not matter much if the bang-bang controller is based on predictive or reactive measurements. The reason for why this parameter affects the system so differently in different simulation setting is hard to understand properly and a brutal trial-and-error approach that tests as many control parameter combinations as possible is therefore the only way to make sure that the controller one sets up will generate acceptable system performance.

[124] Section 4.1.4.

5 Conclusion

5.1 Automatic feedback control in microscopic traffic simulations

The main focus of this master thesis project has been to investigate how well route guidance based on automatic feedback control is able to establish and maintain Nash equilibrium traffic conditions in the microscopic traffic simulation of Berlin called MATSim. The conclusion of the investigation is that that traffic control based on feedback improves the traffic conditions considerably in simulation scenarios where some roads have limited capacity due to traffic accidents.

Different control strategies have been shown to produce differently desirable results in different settings. In a small test network, proportional and integral control produced the best results whereas when the traffic on two roads in the western suburbs of Berlin was controlled, proportional control worked better. Bang-bang control typically generated slightly worse system performance than the two other control strategies, although it is plausible that the bang-bang control is more robust against parameter variations since it is not specifically tuned for any system setting. The use of predictive measurements as an alternative basis for the control output has been shown to improve the system performance for some controllers in some simulation settings. No general conclusions can however be drawn about when to use such measurements. Furthermore, it has been shown that the number of binary messages in the sequence approximating the continuous control signal is an important parameter that should be set to an odd number. It is also advisable to use a dead-zone in the actuation process. Finally, different control strategies have been shown to be differently robust against changes in system parameters. Bang-bang control generates for example relatively better system performances when the time lags are short and P-control is much worse than the other control strategies when the compliance rate is decreased.

Recalling from the introduction that the Berlin MATSim is very complex in terms of network topology and population and that no differential equation model of the simulation is available, there are reasons to assume it to be a lot more difficult to design a working controller in MATSim than in a simplified macroscopic simulation tool where a differential equation is readily available. On the other hand, the higher complexity of the microscopic simulation makes it reasonable to assume the discrepancy between the simulation and the real-world traffic system to be smaller when the simulation is microscopic. Presumably, a controller that improves the traffic conditions considerably in a microscopic simulation would therefore more likely produce acceptable results also if it was installed in the real world than a controller that has been designed within a smooth macroscopic differential equation framework. The controllers designed to handle the traffic in the “Accident scenario” in the reduced Berlin network are indeed such microscopically designed controllers. These controllers improve the traffic condition considerably and one could thus presume their abilities to handle a similar accident on the real highway in the western outskirts of Berlin to be good as well. Nevertheless, if one wants to design a control that can handle more general disturbances on this highway, a controller designed to handle one specific

accident scenario is not sufficient. In the next chapter, a broad outline for a systematic design approach that has a potential of being applicable in a general traffic situation is therefore laid down.

5.2 Setting up a controller in the real world

A prerequisite for controlling a real-world traffic system in the same way as the traffic in MATSim has been controlled within this master thesis project is of course that there exists a real infrastructure for guidance in the system. In this report, the guidance technology has been assumed to be in-car COOPERS-devices installed in every vehicle. As mentioned in Section 3.2.1, the guidance software could as well be said to represent guidance carried out by means of variable direction signs (VDS). The installation of such signs that are directed by a traffic management central would therefore also be sufficient.[125] Another necessary requirement for setting up a real-world control is that the traffic system can be accurately described by MATSim. This may require a substantial calibration work as well as information about the network and the travel pattern of the population in the specific area.

Assuming that these two prerequisites are fulfilled, a first step for setting up a real-world control is to specify in MATSim the road couples that the controllers are to control. As mentioned, control can be carried out simultaneously on arbitrary many couples of routes in the network. In the remaining parts of this section, it is however assumed for simplicity that only one controller is applied in the system. The extension to multiple control is straightforward as long as the controlled areas are sufficiently well spread out in the traffic network.[126] Furthermore, a full MATSim-specification of the control area requires, apart from the two routes, a set of sign-links, a set of destination-links and the sequences of links connecting the sign-links and destination-links with the two routes.[127]

When the control area is fully specified, a set of reasonable problem-scenarios (i.e. accidents, construction works etc.) is to be specified. Depending on the amount of recourses available for the tuning of the controller, this specification could be more or less detailed. Each scenario is defined by its location and the capacity reduction it brings about. The result of the specification is a table like the one below for each of the

---

[125] If the cars are not provided by COOPERS-devices, it would however not be obvious how to measure reactive travel times. This is however a minor problem, since all experiments presented in this report show that the use of predictive travel times as control output give equally good or better system performances.

[126] How much the control areas must be spread out is different in different traffic networks. The important point is that one controller should not make too big of an impact on the system behavior on a route couple controlled by a different controller. That is, the controllers must be somewhat independent of each other.

[127] For details, see Section 3.2.2.

two routes.[128] If there is a lot of time available for control tuning, the amount of different capacity reductions tested can be increased.

Capacity reduction	**20 %**	**40 %**	**60 %**	**80 %**	**100 %**
Link 1					
Link 2					
Link 3					
Link 4					
Link 5					

Table 6. Specification of different problem scenarios on one route in the network.

Thereafter, a reasonable compliance rate for guidance along the two specified routes in the real-world system is to be estimated. This compliance rate is used in the simulations of the network. Moreover, it is also important to somehow estimate how accurate this compliance rate estimation is. This accuracy of the estimation is later used when the control strategies are chosen. At this stage, it is also possible to measure how large a fraction of the agent that uses route 1 when no guidance is applied. This fraction obviously specifies the parameter `nominalSplitting` in the simulation software.

Controllers with optimally tuned parameters are thereafter designed for each scenario in Table 6. If the compliance rate estimate is considered to be accurate, bang-bang control, P-control and PI-control are considered. If the compliance rate is considered to be hard to estimate, P-control is omitted due to its poor robustness against compliance rate variations. For the bang-bang control, only the measurement mode (reactive or predictive) must be decided. One can therefore easily evaluate how this control strategy works in the different scenarios by means of running two sets of bang-bang controlled simulations, one with predictive and one with reactive measurements, and averaging the average disbenefit values (AD) achieved. The smaller of these two *AD*-values is later to be compared to corresponding values achieved with other control strategies.

For the P- and PI-controller a set of parameters are to be tuned for each scenario. The tuning is carried out by means of the trial-and-error approach discussed in Section 4.1.1. As mentioned in Chapter 4.2, this way of tuning the parameters is a computationally costly procedure if the network is large. Depending on the amount of resources, the tuning should therefore be carried out more or less thoroughly. The very best tuning is obviously achieved if all relevant parameters combinations of all relevant parameters are tested. For a PI-controller, there is five degrees of freedom (K_P , T_i, predictive or reactive measurements, `controlEvents` and `deadZone`) and the number of combinations is thus really high even though only a few different values are tested for each parameter. In practice, only a subset of the parameter combinations will thus be tested in the tuning process. Some parameters, considered less important, may

[128] If one of the routes is a main route where all or almost all agents travel (like in the scenarios discussed in Chapter 4.2), problems on the smaller of the two routes are not relevant for the controller. Hence only a table for the main route has to be specified.

for example be tuned roughly initially whereas other more decisive parameters are tuned more carefully later on with the initially tuned parameters kept constant.[129]

For each scenario, the *AD*-value from the simulation controlled by bang-bang control is later compared with the corresponding values achieved when the optimal PI- and, if relevant, P-controller is applied. The controller that generates the best system performance (i.e. the smallest *AD*-value) is later chosen as the most suitable controller for that specific scenario. In scenarios where the differences between the bang-bang control and any other control strategy is non-significant, bang-bang control should always be chosen due to its better robustness against accident size variations. The most suitable control strategy and the parameter setting found optimal is thereafter filled into the appropriate field in the tuning table. An example of how the table thereafter might look is found below.

Capacity reduction	**20 %**	**40 %**	**60 %**	**80 %**	**100 %**
Link 1	BB, react	P, react Kp=0.002 C=3 dZ=0.01	BB, pred	PI, pred Kp=0.005 Ti=1500 C=3 dZ=0.02	BB, react
Link 2	PI, pred Kp=0.0015 Ti=1000 C=5 dZ=0.04	P, react Kp=0.002 C=3 dZ=0.02	P, pred Kp=0.002 C=5 dZ=0.04	BB, pred	PI, react Kp=0.008 Ti=2500 C=7 dZ=0.02
Link 3	PI, react Kp=0.002 Ti=1700 C=5 dZ=0.02	PI, pred Kp=0.002 Ti=1500 C=3 dZ=0.01	BB, pred	P, pred Kp=0.007 C=3 dZ=0.03	PI, pred Kp=0.009 Ti=1500 C=5 dZ=0.02
Link 4	BB, pred	P, pred Kp=0.002 C=3 dZ=0.02	PI, pred Kp=0.002 Ti=1500 C=3 dZ=0.02	PI, pred Kp=0.006 Ti=1500 C=3 dZ=0.01	BB, pred
Link 5	PI, pred Kp=0.002 Ti=2500 C=3 dZ=0.01	PI, pred Kp=0.002 Ti=2000 C=7 dZ=0.02	P, pred Kp=0.002 C=3 dZ=0.01	PI, pred Kp=0.009 Ti=1000 C=3 dZ=0.02	P, pred Kp=0.011 C=3 dZ=0.02

Table 7. Typical tuning table for a route with five links. The existence of fields with P-control indicates that the compliance rate estimation is considered to be comparably accurate.

In addition to the controllers tuned for the different problem settings, a control strategy and a set of parameters that are optimal in a normal day scenario without any capacity reductions should also be tuned in the same way. Remembering how controllers without dead-zone had a tendency of distorting good traffic behavior in the disturbed normal day scenario studied in Section 4.1.2, it is of great importance that the normal

[129] This approach was basically the one used when the parameters of P-controller in the small test network were tuned in Section 4.1.1.

day controller has a sufficiently large dead-zone. It is however not advisable to have no control at all in a normal day scenario in the real world even though that might be indicated by the simulations. In contrast to what is shown in the simulation, there are often great fluctuations in the real-world traffic demand and it is therefore presumably reasonable to have some kind of control turned on at all times.

The tuning procedure described above is a computationally costly procedure that takes a lot of time to carry out. Especially if the routes are long, the number of scenarios as well as the number of simulation runs needed, will be rather large. Nevertheless, when the tuning is finished one has a parameter setting for each possible scenario that is optimal, at least in MATSim. Hence it follows, that the guidance control will be able to cope with any kind of accident or road construction on any link on the routes being controlled. The only thing that the traffic management has to do is to observe where there is a disturbance and estimate how much the capacity is decreased by it. Given these two variables the guidance (may it be transferred via COOPERS or VDS) will be directed optimally by means of the appropriate controller specified in Table 7. When the traffic is not disturbed in any way, the controller found optimal in a normal day scenario is obviously applied.

Finally, it is important stress that a controller that is found to be optimal in MATSim does not necessary control the traffic in the real world as good as it did in the simulation. In order to obtain really good system performance in the real word, real world evaluation measurements and output should be compared with the corresponding measurements from the simulations. If real world measurements are significantly worse than the evaluation measurements obtained with the same controller in the simulation, it might be reasonable to make slight adjustments of the control parameters. If one is lucky, this fine tuning might improve the real world system performance; if not, the parameters from the tuning-table should once again be applied.

5.3 Suggested future research

Following the procedure outlined in the preceding chapter of the report, it should be possible to design guidance system based on automatic feedback control that works good or at least acceptable at an arbitrary location in an arbitrary traffic system in the real world. In the procedure presented, all relevant findings from the simulations presented in Chapter 4 were taken into consideration. The methods for controlling the traffic used in the simulations in Chapter 4 are however not the only possible ways of controlling a traffic system. A lot of design choices were made before the simulations were run, partly due to limitations in MATSim, partly due to lack of time, and partly due to lack of insight of the nature of the system controlled. In this last chapter of this master thesis, a few such alternative design choices will be discussed. Investigations of the value or usefulness of these alterative approaches constitute interesting fields of research for the future.

One major limitation of a controller that is optimized using the tuning proposed in this thesis is that the population simulated behaves more or less identical in each and every simulation run. Even though some randomness is added in the software that handles the

guidance, the major traffic patterns are equal every time the simulation is run. I.e. the same amount of cars is making the same trip from the same origin to the same destination in every simulation as long as they are not affected by the guidance. Obviously this is not a very realistic representation of the traffic patterns in Berlin which of course changes considerably from one day to the other. The controllers optimized in this way thus run the risk of working well only for the very specific traffic pattern studied.

In order to make the controllers more robust against varying traffic patterns, it would therefore be good if the tuning simulations were carried out on populations with slightly different plans in the different simulations. It is however important to notice that the generation of plans files is a computationally costly process. It is therefore not possible to generate one new plans file for every simulation run in the tuning process. An alternative approach that should lead to very similar results is to alter the behavior of the agents in the different simulation runs by adding a random component to the departure time for each leg. In this way, the traffic patterns will be different in every simulation although the same plans file will be used every time round.

Another alternative design approach that has a potential of improving the system performance is to use the disbenefit values $d(t)$ instead of the Nash deviation $y(t)$ as the control output. Recalling that $d(t)$ and not $y(t)$ was the most relevant signal to study when the larger network was simulated, this way of changing the output seems like a reasonable modification of the control design. Presumably, a controller that bases decisions on $d(t)$ would pay attention to the number of agents on each route in a way that is not possible for the controllers discussed in this master thesis. In fact, it is hard to find many good arguments for the use of $y(t)$ as control output. The reason for why $y(t)$ was chosen as the output from the system was quite simply that it is the signal that is used as the system output by Papageorgiou and his research group[130] and it was therefore a design decision made early in the project. The alternative signal $d(t)$ was not even taken into consideration at this early stage and when the focus was shifted towards this signal, it was already way too late to fundamentally change the system output. One relevant reason for why $y(t)$ might be a better system output than $d(t)$ can however be found when a real-system is to be controlled. Contrary to $d(t)$ the number of agents traveling on each route between one of the sign-links and one of the destination-links is not needed to compute $y(t)$. This quantity is easy to obtain if all cars are equipped with COOPERS-devices but in a situation where the guidance is carried out by means of variable direction signs, the number of agents using each route is not trivial to measure. The use of $d(t)$ instead of $y(t)$ as control output therefore has a potential of improving the control and simulations in which this alternative output is used would certainly be very interesting to evaluate. In a system where the control is to be carried out but means of VDS, it is however questionable if such a control output can be used in real world implementations.

[130] Messmer (1994), Papageorgiou (2003)

Finally, it would also be interesting to investigate if bang-bang controllers could be improved by means of adding hysteresis or a dead-zone. In a bang-bang controller with hysteresis, the input generated by the controller does not change sign immediately as the control output changes sign. Instead the control signal is kept constant until the output leaves a pre-defined interval. A similar adjustment of the bang-bang controller would be to add a dead-zone interval for the output.[131] The only difference between hysteresis and a dead-zone is that whereas the control signal keeps its former value when the output is in the critical interval and hysteresis is applied, no guidance at all is given when the output is in the dead-zone.

Hysteresis has been shown to improve system performance in systems controlled by bang-bang control. If the hysteresis interval is chosen properly, the amplitude of the oscillations could sometimes decrease substantially.[132] Seeing that the implementation of a dead-zone resembles hysteresis it is thus reasonable to believe that a dead-zone could have the same kind of effect. Investigations of how the ability of the bang-bang controller to control a traffic system changes when hysteresis or a dead-zone of different size is added would thus have a potential of being very interesting.

---

[131] It is important to notice that the dead-zones used so far in this report have not been applied to the output but to the control signal. I.e. they have been located after the controller, not before it. Such a dead-zone is not able to change the behavior of the bang-bang controller, since the output is always 0 or 1. Some adjustments in the software would thus be needed if one wanted to provide also the bang-bang controller with a dead-zone.

[132] "Hysteresis", Wikipedia; The Free Encyclopaedia
http://en.wikipedia.org/wiki/Hysteresis#Applications (2007-01-13)

References

Published Sources

Balmer, M., Cetin, N., Nagel, K., Raney, B., "Towards truly agent-based traffic and mobility simulations" in *Workshop on agents in traffic and transportation at Autonomous agents and multi-agent systems (AAMAS'04)* (2004)

Bhouri, N., Papageorgiou, M., Blosseville, J. M., "Optimal control of traffic flow on periurban ringways with application to the Boulevard Périphérique in Paris", in *Preprints of the 11th IFAC World Congress, vol 10* (1990), pp 236-243

Daganzo, C. F., "The Cell transmission model: A dynamic representation of highway traffic consistent with hydrodynamic theory" *Transportation Research Part B, vol. 28B, No. 4* (1994), pp 269-278

Daganzo, C. F., "The Cell transmission model, part II: Network traffic" *Transportation Research Part B, vol. 29B, No. 2* (1995), pp 79-93

Diakaki, C., Papageorgiou, M., McLean, T., "Applying integrating corridor control in Glasgow" in *Proceedings of ASCE Fifth International Conference on Application of Advanced Technologies in Transportation Engineering* (1998), p 1-16

Elloumi, N., Haj-Salem, H., Papageorgiou, M., "METACOR: a macroscopic modelling tool for urban corridors" *Proceedings of TRISTAN II, vol. 1* (1994), p 135-150

Flötteröd, G. & Nagel, K., "Some Practical Extensions to the Cell Transmission Model" Conference Paper from IEEE ITSC, Vienna, 2005

Glad, T. & Ljung, L., *Reglerteknik; Grundläggande teori* 2nd edition (Lund, 1989)

Glad, T. & Ljung, L., *Reglerteori; Flervariabla och olinjära metoder* 2nd edition (Lund, 2003)

Kerner, B. & Rehborn, H., "Experimental properties of complexity in traffic flow" *Phy. Rev. E, 5*, (1996)

Kotsialos, A., Papageorgiou, M., Mangeas, M., Haj-Salem, H., "Coordinated and integrated control of motorway networks via non-linear optimal control" *Transportation Research Part C 10* (2002), pp 65-84

Mammar, S., Messmer, A., Jensen, P., Papageorgiou, M., Haj-Salem, H., Jensen, L., "Automatic control of variable message signs in Aalborg" *Transportation Research Part C, vol. 4, No. 3* (1996), pp 131-150

Messmer, A. & Papageorgiou, M., "METANET: a macroscopic simulation program for motorway networks" *Traffic Engineering and Control 31* (1990a), pp 466-470

Messmer, A. & Papageorgiou, M., "Route diversion control in motorway networks via nonlinear optimisation" *IEEE Transactions and Control Systems Technology 3* (1990b), pp 144-154

Messmer, A. & Papageorgiou, M., "Automatic control methods applied to freeway network traffic" *Automata Vol. 30, No. 4* (1994), pp 691-712

Nagel, K., Rickert, M., Simon, K. M., Pieck, M., "The dynamics of iterated transportation simulation" (2000), in *Preprints of the TRISTAN-3 Conference, San Juan, Puerto Rico, 1998*

Papageorgiou, M., "Traffic Control" in Hall, R.W., ed. *Handbook of Transportation Science* 2nd edition (New York, 2003)

Råde, L. & Westergren, B., *Mathematics Handbook for Science and Engineering* 4th Edition (Lund, 2001)

Wang, Y., Papageorgiou, M., Messmer, A., "A predictive feedback feedback routing control strategy for freeway network traffic" for the Transportation Research Board. 82nd Annual Meeting, January 12-16, 2003, Washington, D. C.

Internet Sources

The COOPERS Website: http://www.coopers-ip.eu (2007-01-08)

Flötteröd, G. & Nagel, K., "Simulation and optimization of trajectories in a congested network" (2006) published on the Institute for Transport System Planning and Transport Telematics, TU Berlin website: http://www.vsp.tu-berlin.de/archive/sim-archive/publications (2007-01-26)

"Hysteresis", Wikipedia; The Free Encyclopaedia http://en.wikipedia.org/wiki/Hysteresis#Applications (2007-01-13)

Martin, R. C., "UML Tutorials: Part 1 – Class Diagrams" published on the Object Mentor website: http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf (2007-01-13)

"Nash equilibrium", Wikipedia; The Free Encyclopaedia http://en.wikipedia.org/wiki/Nash_equilibrium (2007-01-08)

Simon, P. M., Esser, J., Nagel, K., "Simple queuing model applied to the city of Portland" (1999) published on the Institute for Transport System Planning and Transport Telematics, TU Berlin website: http://www.vsp.tu-berlin.de/archive/sim-archive/publications (2007-01-08)

Appendices

A. The small test network - link specification

Link Id	Length [m]	Capacity [cars/hour]	Free flow speed [m/s]
1	500	3000	27.78
2	500	6000	27.78
5	2000	3500	27.78
6	30	1300/1900	27.78
7	2000	3500	27.78
8	30	1900	27.78
14	500	3000	27.78
16	500	3000	27.78
20	1000	6000	27.78
21	1000	6000	27.78
22	3500	3000	27.78
23	1000	3000	27.78

The capacity on link 6 is different in the "Normal day" and "Accident" scenario respectively.

B. The small Test network – simulation results

Normal day

	AN	*NM*	*AD*
No disturbance, no control	0,423	-0,061	3,5
Disturbance, no control	5,03	0,0805	149
Disturbance, BB reactive	33,3	-0,311	1117
Disturbance, BB predictive	19,4	-0,126	531
Disturbance, PI-control, reactive	5	-0,0358	148
Disturbance, PI-control, reactive, no dead-zone	13,4	-0,202	367
Disturbance, PI-control, predictive	5,02	-0,0697	148
Disturbance, PI-control, predictive, no dead-zone	13,3	0,628	371

The disturbance is always normally distributed with a standard deviation of five seconds. The control parameters are Kp=10^-2.8, Ti=10^8/3.

Accident

	AN	*NM*	*AD*
No Control	419	364	80354
All agents guided along route 2	337	-324	75384
BB, reactive	112	39,1	6660
BB, predictive	45,1	18,8	1363
P-control, reactive	28	16,5	927
P, predictive	26,5	16,5	838
P-control, reactive, no dead-zone	44,1	25,7	1675
PI-control, reactive	22,2	2,07	745
PI-control, predictive	21	2,18	665

The control parameters are Kp=10^-2.8, Ti=10^8/3 for all controllers except for the P-controller without dead-zone. For that controller, a specific tuning was done and the control parameter Kp=10^-2.4 was picked.

C. The reduced Berlin network – simulation results

Normal day

	AN	*NM*	*AD*
No disturbance, no control	224	204	0
Disturbance, no control	223	203	0
Disturbance, BB reactive	222	201	0
Disturbance, BB predictive	225	200	0
Disturbance, PI-control, reactive	225	205	0
Disturbance, PI-control, predictive	226	-203	0

The control parameters are for the P-control Kp=10^-2.5 and for the PI-controller, Kp=10^-2.67 and Ti=100

Accident

	AN	*NM*	*AD*
No Control	151	51.6	1959
All agents guided along route 1	681	593	30,661
BB, reactive	144	126	274
BB, predictive	150	137	319
P-control, reactive	125	90	228
P, predictive	128	106	143
PI-control, reactive	144	114	515
PI-control, predictive	134	112	231

The control parameters are for the P-control Kp=10^-2.5 and for the PI-controller, Kp=10^-2.67 and Ti=100