Look Ma, No Hands!
Aspects of Autonomous Vehicle Control

Willem van Willigen
Department of Computer Science
Faculty of Sciences, Vrije Universiteit
2014
Thesis Reading Committee:

- dr. T. Bosse, Agent Systems Research Group, Faculty of Sciences, VU University Amsterdam, The Netherlands
- prof.dr.ir. S.P. Hoogendoorn, Transport & Planning Department, Faculty of Civil Engineering and Geosciences, Delft Technical University, The Netherlands
- prof.dr.ir. J.W.C. van Lint, Transport & Planning Department, Faculty of Civil Engineering and Geosciences, Delft Technical University, The Netherlands
- dr. S.K. Smit, Modelling & Simulation Department, TNO The Hague, The Netherlands
- dr. A. Visser, Informatics Institute, Faculty of Science, University of Amsterdam, The Netherlands

The research reported in this thesis was supported by the Distributed Sensor Systems group at TNO and its program on Adaptive Multi-Sensor Networks (AMSN).

SIKS Dissertation Series No. 2014-12
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Cover Design: Evert Haasdijk, Daan Krikke and Willem van Willigen.
Copyright © 2014 by W.H. van Willigen. All rights reserved.
Look Ma, No Hands!
Aspects of Autonomous Vehicle Control

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. F.A. van der Duyn Schouten,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Exacte Wetenschappen
op woensdag 28 mei 2014 om 11.45 uur
in de aula van de universiteit,
De Boelelaan 1105

doort

Willem Hotze van Willigen

geboren te Amsterdam
promotor: prof.dr. A.E. Eiben

copromotoren: dr. E.W. Haasdijk
dr.ir. L.J.H.M. Kester
# Table of Contents

Dankwoord ................................................................. v

1 Summary .................................................................. 1

2 Introduction ................................................................ 3
  2.1 Motivation ............................................................... 3
  2.2 Objectives and Contributions .................................... 5
  2.3 Structure of the Thesis .............................................. 6
  2.4 Publications and Research Overview ......................... 8

I Autonomous Flying ...................................................... 11

3 Autonomous UAV Path Planning .................................. 13
  3.1 Introduction ............................................................ 15
  3.2 Model .................................................................. 16
    3.2.1 Problem setting ................................................. 17
    3.2.2 Solution Approach ............................................ 19
    3.2.3 UAV adaptive agent .......................................... 20
  3.3 Experiments ........................................................... 21
    3.3.1 Design and Setup ............................................. 21
    3.3.2 Results .......................................................... 22
    3.3.3 Analysis ........................................................ 23
  3.4 Conclusions .......................................................... 25

II Autonomous Driving ................................................... 27

4 Safety in the Face of Uncertainty ................................. 43
  4.1 Evaluating Adaptive Cruise Control Strategies in Worst-Case Scenarios ........................................... 45
    4.1.1 Introduction ..................................................... 45
    4.1.2 Related Work .................................................. 47
    4.1.3 Model ............................................................. 48
    4.1.4 Experiments and Results .................................... 53
    4.1.5 Analysis ........................................................ 56
4.2 Approximating Safe Spacing Policies for Adaptive Cruise Control Strategies ................................................. 59
  4.2.1 Introduction ................................................................. 59
  4.2.2 Background ................................................................. 61
  4.2.3 Model .................................................................. 63
  4.2.4 Experiments and Results ........................................... 68
  4.2.5 Analysis .................................................................. 71
  4.2.6 Conclusion ................................................................. 74

4.3 Critical Headway Estimation in Cooperative Adaptive Cruise Control ................................................................. 77
  4.3.1 Introduction ................................................................. 77
  4.3.2 Related Work ............................................................... 79
  4.3.3 Method .................................................................. 81
  4.3.4 Validation .................................................................. 92
  4.3.5 Conclusion ................................................................. 97

5 Preference-based Controller Building ................................................. 99
  5.1 Evolving Intelligent Vehicle Control using Multi-Objective NEAT ................................................................. 101
    5.1.1 Introduction ................................................................. 101
    5.1.2 Related Work ............................................................... 104
    5.1.3 Algorithm ................................................................. 105
    5.1.4 Experiments ................................................................. 110
    5.1.5 Results and Analysis .................................................. 113
    5.1.6 Conclusions and Future Work ..................................... 117

  5.2 A Multi-Objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems ................................................................. 118
    5.2.1 Introduction ................................................................. 118
    5.2.2 Related Work ............................................................... 121
    5.2.3 Algorithm ................................................................. 123
    5.2.4 Experimental Setup .................................................... 124
    5.2.5 Results and Analysis .................................................. 130
    5.2.6 Conclusions and Future Work ..................................... 133

  5.3 Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives ................................................................. 135
    5.3.1 Introduction ................................................................. 135
    5.3.2 Related Work ............................................................... 138
    5.3.3 Algorithm ................................................................. 140
    5.3.4 Experimental Set-Up .................................................... 141
    5.3.5 Results and Analysis .................................................. 148
    5.3.6 Conclusions and Future Work ..................................... 149

  5.4 The Fast and the Comfortable: Evolving Preference-Based Controllers for Autonomous Vehicles ................................................................. 153
    5.4.1 Introduction ................................................................. 153
    5.4.2 Related Work ............................................................... 156
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.3</td>
<td>Controller</td>
<td>157</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Experimental Set-Up</td>
<td>162</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Results and Analysis</td>
<td>166</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Conclusion</td>
<td>173</td>
</tr>
<tr>
<td>III</td>
<td>Multi-Objectivation</td>
<td>177</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>180</td>
</tr>
<tr>
<td>6.2</td>
<td>Background</td>
<td>181</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Non-Dominated Sorting</td>
<td>182</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Pareto Strength</td>
<td>183</td>
</tr>
<tr>
<td>6.3</td>
<td>Extending CMA-ES</td>
<td>183</td>
</tr>
<tr>
<td>6.4</td>
<td>Experiments and Results</td>
<td>185</td>
</tr>
<tr>
<td>6.5</td>
<td>Discussion</td>
<td>189</td>
</tr>
<tr>
<td>6.6</td>
<td>Conclusion</td>
<td>191</td>
</tr>
<tr>
<td>7.1</td>
<td>Look Ma, No Hands!</td>
<td>193</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Autonomous UAVs</td>
<td>194</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Safety in the Face of Uncertainty</td>
<td>194</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Customisability by Preference-Based Controller Building</td>
<td>195</td>
</tr>
<tr>
<td>7.2</td>
<td>Contributions of the Thesis</td>
<td>195</td>
</tr>
<tr>
<td>7.3</td>
<td>Future Research</td>
<td>196</td>
</tr>
</tbody>
</table>

**References**

**Summary in Dutch: Kijk mam, zonder handen!**
Opdracht volbracht. De zaak is rond.

Harry Mulisch, De ontdekking van de hemel

Dankwoord

Het proefschrift dat voor u ligt markeert het einde van een fantastische vier jaar van mijn leven. Het is geweldige mijlpaal, en het is een bizar idee om deze tijd nu definitief achter me te laten. Alleen had ik het niet gekund – mijn dank gaat uit naar de volgende mensen.

Papa, mama, zonder jullie had ik nooit zover kunnen komen. Dank jullie wel voor de verjaardagspartijtjes naar het NINT museum. Ik heb het vermoeden dat daar mijn interesse in de wetenschap is ontstaan. Ik heb me altijd onvoorwaardelijk gesteund gevoeld door jullie.

Martijn, bij jou is het allemaal begonnen. Je hebt mij bij TNO geïntroduceerd toen ik op zoek was naar een afstudeerstage, en mede dankzij jouw inspanningen heeft mijn promotietraject uiteindelijk kunnen beginnen. Zonder jou had dit proefschrift er dus nu niet gelegen. Ook aan de werkzaamheden die we samen hebben gedaan voor het Aware project denk ik met veel genoegen terug.

Gusz, je was een inspirerende promotor. Toen ik voor het eerst bij je op de kamer binnen kwam in mijn Dark Side of the Moon T-shirt en jij daar gelijk vol enthousiasme op reageerde, wist ik dat ik een leuke baas te pakken had. Je verstaat de kunst om als begeleider binnen een paar zinnen de vinger op de zere plek te leggen, zelfs al had ik je al een tijd niet inhoudelijk gesproken (of kwam het juist daardoor?)? Onze reisjes naar Edinburgh en Singapore waren natuurlijk fantastisch, en daar kijk ik met veel plezier op terug.

Leon, dank voor je begeleiding vanuit TNO. Ik heb genoten van onze discussies over zeer uiteenlopende onderwerpen. Je hebt me een hoop geleerd, en niet in de minste plaats over de filosofie van de wetenschap, waar je helden Spinoza en Popper vaak de revue passeerden. Ik kijk uit naar het boek dat je gaat schrijven!

Evert, het was een feest om jou als begeleider te hebben. Zowel inhoudelijk als persoonlijk hebben we een goede klik. Dank je wel voor alle tijd en aandacht die je aan mijn proefschrift heb besteed. Natuurlijk ook voor de prachtige foto op de kaft. Dat we nog maar vaak bier mogen drinken en naar Zappa plays Zappa mogen gaan.

Dank aan de leden van de leescommissie, voor jullie tijd en feedback: Tibor Bosse, Serge Hoogendoorn, Hans van Lint, Selmar Smit en Arnoud Visser.

Dank aan al mijn (oud-)VU-collega’s van de CI groep: Rob Konijn, Paul van Klaveren, Vincent van der Goes, Eelco den Heijer, Luís Simões, Giorgos Karafotias, Berend Weel, Nivea Schut-Ferreira, Mark Hoogendoorn, Wojtek Kowalczyk, en
Zoli Szlávik. Ik mis de sloten koffie en de raadsels, de uitjes en de wekelijkse (of iets minder wekelijkse) update-sessies. Ook dank aan mijn TNO-collega’s André Verweij, Peter Hiemstra, Jeroen Bergmans, Eelke van Foeken, Harm van Seijen, Jan-Willem Marck, Martin van Rijn, Ellen van Nunen en Jeroen Ploeg.

Paranimf Mark, dank je wel. Voor je morele ondersteuning, voor de niet geringe hoeveelheid kipsaté die we met elkaar naar binnen hebben gewerkt, voor alle squashwedstrijden, Nieuwpoortparty’s en de pubquizvoorbereidingen. Om onze favoriete film te citeren: I am overjoyed!

Paranimf en broer Rutger, wat ben je toch een koning. Je was een van de weinigen uit mijn privé-omgeving die ook daadwerkelijk begreep waarover ik het had als ik met een inhoudelijk probleem zat, en je had altijd tijd (of beter gezegd: je maakte altijd tijd) om even koffie te drinken om een kleine dosis welschmerz door te nemen. Dank je wel daarvoor. Ik ben trots op je.

Annelies, Danny, dank jullie wel voor de twee liefste nichtjes. Visto, dank voor de kerstdiners en de talloze avonden in de Koe. Butú’s en butú’s, dank voor de Ajaxmiddagen en de filmsessies (en dan natuurlijk vooral de bloopers). Familie de Bruijne, dank voor de steun en de gezelligheid van de afgelopen jaren. Daan, dank voor je werk aan de kaft. Deloitters, dank dat ik me gelijk bij jullie thuis ben gaan voelen, en voor de tijd die jullie me gunden om nog aan mijn proefschrift te werken.

Laura, jou heb ik voor het laatst bewaard. Al vier jaar lang deel ik alles met je. Je weet als geen ander dat de afgelopen vier jaar mij niet altijd even gemakkelijk zijn gevallen. Jij verstaat de kunst om mij te laten relativeren, op de momenten dat ik zelf ervan overtuigd ben dat alles in de soep loopt. Je bent de liefde van mijn leven en ik geniet van elk moment met je.
Summary

Whether we are on land, at sea or in the air: we are surrounded by transportation systems. An ever-growing world population and further globalisation increases the dependency of humankind on efficient transportation methods.

This thesis is about autonomous vehicles, which means that there are no human pilots or drivers involved in the control of the vehicle. Examples of relevant questions within this topic are: How do you plan the route of an autonomous airplane? How can you guarantee that an autonomous car brakes in time? How can an autonomous vehicle adopt a certain driving style, following the driver’s preference? These are the type of questions that this thesis considers.

The thesis is divided into three parts. The first part is about autonomy in unmanned aerial vehicles. When there is a trade-off to be made between exploration (see as much as possible) and exploitation (identify as much as possible) of some unknown terrain, it is possible to make the plane adapt its flight path according to its observations while flying. If the plane encounters an interesting object, it can decide to circle above this object for some time, before continuing its predetermined flight path. The pre-determined flight path is found using a Particle Swarm Optimiser.
The second part is the main part of the thesis. Its topic is self-driving cars on highways. There is an ongoing trend towards intelligent transportation systems, where cars and the infrastructure become more and more autonomous. Sensors such as radar, GPS and Wi-Fi can measure state information about the vehicle and its surrounding area, and by using this information, the autonomous vehicle ‘knows’ what the minimum safe distance is that it should keep. An important aspect here is safety. Sensors are not perfect: radars have uncertainty in their measurements, Wi-Fi communication can drop. This thesis describes a method that can compute the minimum safe distance between two vehicles, given the uncertainty of the vehicle’s sensors, vehicle behaviour and communication system.

A more futuristic chapter of this thesis is about translating user preferences into vehicle controllers. Some drivers like speed, others prefer comfort or fuel economy. These objectives are potentially conflicting. This chapter describes a method that uses multi-objective evolutionary computing to evolve not one, but a whole range of controllers, that each signify a different prioritisation between a number of user objectives.

In the final part of this thesis, I describe a generalisation of the multi-objective evolutionary algorithm that was introduced in part II. This chapter reports on the performance of this algorithm on a well-known set of multi-objective optimisation problems.

Autonomous vehicles, especially on the road, are an important part of our future. This thesis addresses some important aspects of getting closer to this future. Some contributions in this thesis are relevant in the development of current systems, while other contributions become relevant in a more distant future. But we do not doubt that this future is upon us.
Introduction

2.1 Motivation

Transportation systems can be found everywhere around us; whether we look on land, in the air or on water, there are many people who need to go from one place to the other. With an ever-increasing world population and the ongoing process of globalisation, people become more and more dependent on efficient transportation systems.

There is trend in transportation systems where vehicles become more and more autonomous. Autonomy in vehicles (whether they are cars, airplanes, boats or other means of transportation) has the potential to make transportation systems more efficient and safe. A couple of relevant examples.

− More than 90% of all traffic accidents are due to human error (Olarte, 2011). This costs a lot of lives and money. Automated vehicles are not distracted by phone calls, do not experience fatigue and its sensors have faster response times than humans. These factors all have the potential to reduce the number of traffic accidents.
Chapter 2. Introduction

− Traffic jams are often caused by distracted drivers who brake unnecessarily, the so-called ‘ghost jams’. Also in this case, autonomous vehicles have the potential to reduce this problem.

− In the domain of path planning, humans are notoriously bad at finding optimal routes, especially when there is more than one goal to adhere to, and when the degrees of freedom are high (e.g. in the air). Automating this process can be very beneficial.

− Adaptive Cruise Control is already commercially available, and has merits to traffic throughput (Minderhoud, 1999; VanderWerf et al., 2002; Van Arem et al., 2006).

It is a generally accepted fact that computers are much better than humans in solving tedious and repetitive tasks. For example, autopilots in airplanes take over the control from the human pilots for almost the entire flight, except the take-off and landing. As a result, we envision transportation systems of the future as being highly autonomous. Cars, airplanes, boats, any kind of vehicle will have a certain degree of autonomy.

When designing and implementing such autonomous systems within the transportation domain, there are two major goals for modern innovations:

Efficiency Increasing the maximum throughput on highways (i.e. the maximum number of vehicles that move through a stretch of road over a period of time) and the resulting increased efficiency is an important goal for modern transportation systems. Traffic congestion remains a big problem world-wide and modern systems try to reduce this problem, thereby saving money and time.

Safety Innovations in the transportation domain are only acceptable when the resulting systems are safe. Therefore, safety is a primary goal for new systems. Furthermore, in the current situation, things are far from safe: the World Health Organisation has estimated that in 2013, there were 1.24 million traffic deaths around the world (World Health Organisation, 2013).

Creating autonomy in systems implies that people should give up tasks that they are currently used to doing. One of the great challenges of removing people from such tasks, is to have these people accepting and trusting these systems.
Apart from safety and efficiency, that already have been mentioned as primary objectives, we identify the following two constraints for the acceptance of autonomous systems:

**Customisability** Different users of autonomous systems can have different goals or objectives. Users would adopt autonomy faster when they can customise it to their particular wishes.

**Affordability** The systems that we envision must be affordable. For example, autonomous vehicles such as Google’s self-driving car work well, but all the technology that they put into their vehicle is still very expensive. Drivers will only accept autonomy in their vehicles if they can afford it. Affordability is also important when governments get involved: investing in autonomous systems can be a very attractive alternative to building new roads.

We primarily look for solutions in the field of computational intelligence, a relatively new subfield of computer science. Many algorithms within this field take their inspiration from nature. These algorithms are known to be robust and they are able to adapt to changing circumstances. This robustness and adaptivity means that such algorithms are well-suited for problems that are so complex that solutions are very hard to be designed and coded by traditional analytical means.

### 2.2 Objectives and Contributions

From the motivation described in the previous section, the following objectives for the thesis can be identified:

1. Identify applications within the transportation systems domain where autonomy can be beneficial.

2. Develop algorithms that are useful for these applications and develop demonstrators to test the algorithms.

3. Implement, test and assess the quality of these algorithms and, if possible, compare them to existing benchmark algorithms.

This thesis contains the following scientific contributions. In the domain of path planning for autonomous UAVs, we constructed an algorithm that a) determines
a priori the best flight given some objective function and some prior knowledge of the terrain and b) is able to slightly adjust its flight path based on actual observations, when the UAV is in the air.

We also developed a method that is able to calculate the critical safe headway time between two vehicles on highways, given several factors of uncertainty, in real time. This method is applicable in the domain of cooperative adaptive cruise control, in which vehicles on highways use direct communication and other sensors to drive very closely to each other. To ensure safety in such systems, being able to deal with uncertainty in information, communication and system behaviour is crucial.

Another main contribution is a multi-objective evolutionary algorithm that constructs a set of controllers for autonomous vehicles, each signifying a unique prioritisation between certain driver preferences. This algorithm is applicable within the domain of higher-level autonomous vehicle control on highways. The novelty of the algorithm lies in that it is preference-based. Different drivers have different objectives: some prefer going fast, others prefer comfort or fuel economy. Given a certain set of preferences, a vehicle can select the appropriate controller that adheres best to these preferences.

The final contribution of this thesis is a generalisation of the earlier mentioned multi-objective evolutionary algorithm: we tested if we could multi-objectify another evolutionary algorithm by the same method used in chapter 5. The results of these experiments rejected the hypothesis that our multi-objectivation method is always applicable.

### 2.3 Structure of the Thesis

The thesis is structured in three parts. Part I starts off with chapter 3, that describes our first efforts into the domain of autonomous vehicles. More specifically, this research described in this chapter is about path planning for autonomous unmanned aerial vehicles (UAVs). When the task of a UAV is to explore an unknown terrain, its path needs to be planned in advance. During the flight, however, the UAV may decide autonomously to explore certain locations in more detail, for example, because the UAV detected an object of interest. These on-line changes in flight path are based on a utility function, that predicts how utilile the expected information gained in the remaining flight path is. By comparing the values of
the expected utility function given a) the originally planned flight path and b) the (slightly) changed flight path, a decision can be made. Of course, in this particular application, the definition of the utility function is crucial in the behaviour of the UAV: the utility function can be tuned in many different ways, resulting in different flight behaviours of the UAV.

In part II we switch our focus to road transportation systems. We found that in these systems, there are many more degrees of freedom (more vehicles, more constraints, more involvement of humans). In the introduction of part II, we give an introduction of these intelligent transportation systems and the direction in which the research within this field is headed. We argue that in order to improve safety and efficiency in these systems, the development of autonomous vehicles is imperative. In this chapter, we also address issues such as security, liability and acceptability.

The subject of safety in (cooperative) adaptive cruise control is tackled in chapter 4. In this chapter, we describe a model that has the ability to deal with uncertainty in information, communication and system behaviour in real time.

In chapter 5, we move our focus to a method for creating higher-level controllers of autonomous vehicles on highways. We argue that vehicle owners want highly customisable vehicle controllers and this customisability is a core aspect of our method.

Part III consists of chapter 6, in which we generalise the multi-objective evolutionary algorithm that was used in chapter 5. We multi-objectified another otherwise single-objective evolutionary algorithm and tested how well it solves a number of well-known multi-objective numerical optimisation functions.

Finally, in chapter 7, we draw general conclusions about the research objectives that we stated above. We also provide some general directions of future research that can follow up on the work that is described in this thesis.

It is important to note that the sections of chapters 4 and 5 have been included exactly as they were published. The consecutive papers are based on iterative research, and each paper builds on the previous one. This means that the sections within these chapters have a lot in common – for example, the context, background and parts of the models that were used contain a lot of similarities. Therefore, in the introduction of these chapters, we have included a ‘reading guide’, which explains the continuity between the sections of that chapter.
2.4 Publications and Research Overview

Table 2.1 shows which sections tackle which aforementioned challenges and/or constraints. Chapter 6 does not appear in this table, since the work in this chapter is algorithmic in nature and therefore does not fit in within the domain-specific goals and constraints.

<table>
<thead>
<tr>
<th>Year</th>
<th>Efficiency</th>
<th>Safety</th>
<th>Customisability</th>
<th>Affordability</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>Chapter 3</td>
<td>Section 4.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Section 4.2</td>
<td>Section 4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Section 4.3</td>
<td>Section 4.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Section 5.1</td>
<td>Section 5.1</td>
<td>Section 5.1</td>
<td>Section 5.1</td>
</tr>
<tr>
<td></td>
<td>Section 5.2</td>
<td>Section 5.2</td>
<td>Section 5.2</td>
<td>Section 5.2</td>
</tr>
<tr>
<td></td>
<td>Section 5.3</td>
<td>Section 5.3</td>
<td>Section 5.3</td>
<td>Section 5.3</td>
</tr>
<tr>
<td></td>
<td>Section 5.4</td>
<td>Section 5.4</td>
<td>Section 5.4</td>
<td>Section 5.4</td>
</tr>
</tbody>
</table>
This thesis is based on the following papers:


Chapter 2. Introduction


Part I

Autonomous Flying
This chapter describes an autonomous path planning algorithm for unmanned aerial vehicles (UAVs).

When objects in a potentially hostile environment need to be detected and identified, it is a good idea to do this using UAVs, since these planes can fly without people on board and they are relatively cheap as well. In current practice, however, UAVs are generally still operated manually by an operator. This has some very practical downsides: radio communication between the ground and the UAV is still necessary and the human on the ground may not be able to make the best assessment about what to do next, especially when there are multiple objectives to optimise.

We describe a scenario in which a UAV has a limited amount of fuel (i.e. there is a fixed maximum flight path length) and two separate goals: to maximise the number of observed ‘targets’ and to identify these detected targets as well as possible.

This chapter contains the following contributions:

- A particle swarm optimisation (PSO) algorithm for finding initial paths that the UAV will fly. These paths are evolved by the PSO based on some prior
knowledge about the terrain the needs to be explored, such as the height of the terrain and a probability distribution about where the targets most likely will be.

- A on-line algorithm that decides whether or not to stay in a certain location upon detecting a target (thereby effectively shortening its flight path at the end). This decision is based on the outcome of an expected utility function. This algorithm runs while the UAV is flying the path that was determined by the PSO.

These efforts in optimising multiple goals for autonomous vehicles gave us some valuable insights. First, we saw that by using a PSO to optimise our a-priori flight path, the diversity in resulting paths was low. The particles in the swarm each represented waypoints of the flight path and restrictions on these waypoints (e.g., two consecutive waypoints must have a pre-defined distance between them) made it very hard to evolve radically different flight paths. Second, while our method does make a distinction between multiple goals, it did not create a set of optimal solutions in the way a multi-objective algorithm would. By contrast, in chapter 5, we show an example of a multi-objective algorithm that is able to create multiple solutions, that each optimise on a different prioritisation of the objectives.
3.1 Introduction

One of the most prevalent and important issues in reconnaissance, surveillance and target acquisition (RSTA) flight missions is the ability to adapt one’s flight path based on acquired information. In such (often military) missions, planes acquire information about a specific territory by first exploring it, followed by surveilling and finally obtaining information about possible targets in the area. While some information about the territory may be available beforehand (making a priori planning possible), it is increasingly important to do the planning during the mission itself because of the very dynamic nature of RSTA missions at present day (e.g., unknown territory, rapidly moving targets).

The possibility of such automated adaptability during the mission becomes very important when we take the human out of the loop, as we employ unmanned aerial vehicles (UAVs) in RSTA missions. The problem that we address in this section concerns the programming of such UAVs in situations where some information is available beforehand (for example, some knowledge about possible target locations throughout the territory), but where substantial performance may be gained by equipping the UAVs with online (in-flight) adaptation of the flight path based on collected real-time information. We employ a machine-learning approach to accomplish this. Machine learning has been used to deal with different issues in UAV research and development. For example, Berger et al. (2009) use a co-evolutionary algorithm for information gathering in UAV teams; Allaire et al. (2009) have used genetic algorithms for UAV real-time path planning; and Sauter et al. (2005) and Legras et al. (2008) have used a swarming approach (for which a ground sensor network for coordination purposes is needed).

Recently, Pitre (2011) introduced a new measurement for (UAV) search and track missions. The introduced metric jointly optimises the objectives to 1) detect new targets and 2) track previously detected targets. This particular metric has

Chapter 3 was published as:

some desirable properties with respect to search-and-tracking: jointly optimises detection and tracking; easily compares different solutions; promotes early detection; encourages repeated observations of the same targets; and it is useful for resource management. However, this approach does not yet allow for online adaptation of the search path during the flight. In this section, we provide a method for doing this.

We build further on the work of Pitre et al. with two important differences: 1) we use the metric and calculations also for in-flight coordination and adaptation (whereas the original metric has reportedly only been used for off-line generation of paths) and 2) in our case study, the second objective (besides search) is to identify targets rather than tracking these.

This section is structured as follows. In subsection 3.2, we present the details of our adaptive algorithm. We report on the conducted simulation study in subsection 3.3. Finally, subsection 3.4 concludes and provides some pointers for future work.

### 3.2 Model

In this section, we describe the model that we used in terms of (1) the problem setting (i.e., search-and-identification of targets in some terrain with UAVs) and (2) our solution approach (i.e., objective function and adaptive behaviour of the UAV). We describe both these aspects in detail below.

Our solution approach enables a UAV to jointly optimise the objectives of searching and identification by a UAV in a given terrain. Although we have no exact knowledge on where targets are in the terrain (because that would render the search-aspect of the mission pointless), we have some a priori knowledge in terms of probability distributions over the terrain cells on whether a target could be there. Before the mission, we compute an optimal flight path for the UAV. When the UAV is in-flight, it is possible to adapt this path. The before-mission calculation of the optimal search path as well as the in-flight decision to-adapt-or-not is based on a number of value functions that are described in detail below.
3.2. Model

![Figure 3.1 – Scenario Maps (Taken from Pitre et al. (2009)).](image)

![Table 3.1 – Scenario Assumptions.](image)

<table>
<thead>
<tr>
<th>Terrain type</th>
<th>( p_{\text{dot}} )</th>
<th>( p_{\text{dot on road}} )</th>
<th>( % ) Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desert</td>
<td>0.90</td>
<td>0.95</td>
<td>90</td>
</tr>
<tr>
<td>Mountain</td>
<td>0.5</td>
<td>0.75</td>
<td>7</td>
</tr>
<tr>
<td>Forest</td>
<td>0.10</td>
<td>0.50</td>
<td>1</td>
</tr>
</tbody>
</table>

3.2.1 Problem setting

**Terrain**  The terrain to-be-searched is 60 by 60 nautical miles (nmi). This consists of a mountainous area, a desert, a small forest and some roads. In Figures 3.1(a) and 3.1(b), two maps of the terrain show the different types of terrain and the different altitudes (that ranges from 856m to 2833m), respectively\(^i\). In both figures, the straight lines depict roads in the terrain.

A UAV that flies over the terrain cannot detect targets equally well in all types of terrain. We represent the ability-to-detect by means of a detection probability, denoted by \( p_{\text{dot}} \), where dot means detection-on-terrain. In Table 3.1(a), the detection probabilities for the different types of terrain are shown. The right column of this table shows that the detection probabilities increase when targets are on a road.

**Targets**  In this scenario, targets are stationary (i.e., non-mobile) objects located throughout the searched terrain. We consider all targets to be equally important (i.e., not prioritising with respect to a specific aim of a mission)\(^{ii}\). Targets can

---

\(^i\)These maps are the same that were used in Pitre et al. (2009).

\(^{ii}\)In Pitre et al. (2009), extensions are introduced that allow for varying the target importance.
be identified better when they are observed longer. We represent this gradually improving identification by means of a single scalar value, which increases as a UAV observes the object longer.

**UAVs** The UAVs in our model are planes that fly with a constant speed of 100 knots (kt) at a constant altitude of 3,000 meters above sea level. As previously mentioned, the UAV flies a particular search path that was determined beforehand. The adaptability of the UAV is that upon observation of a target, it may decide to fly a circle over the target enabling better identification. This decision depends on the objective function presented later in this section. After finishing the circle, it continues its original search path. A UAV has only limited resources (e.g., fuel), thus when it decides to fly a circle, this means that the path shortens in the tail (details follow below).

How much a UAV can see on the ground, depends on the altitude of the terrain. The detection range is defined as \( \text{range}(alt) = -6.5 \cdot 10^{-4} \cdot alt + 1.96 \), where \( alt \) is the altitude of the terrain. We assume a viewing angle of about 51 deg in every direction. In the lowest regions of the terrain, the detection range is 1.4 nmi, while in the higher regions, this number drops to just 0.1 nmi. Note that the UAV maintains a constant absolute altitude, so when it flies in the mountains, it is closer to the ground.

The probability that a UAV detects a target on the ground, denoted by \( p_{\text{det}}(,) \), is determined by the detection range:

\[
p_{\text{det}}(\text{cell}) = \begin{cases} 
p_{\text{dot}} & \text{if within range(alt)} \\
0 & \text{otherwise} \end{cases}
\]

(3.1)

where \( \text{cell} \) is a single location in the terrain.

The UAV sensor automatically takes a picture every 30 seconds. In our scenario, a mission takes 2 hours, thus resulting in a total of 240 pictures taken and analysed. Finally, the maximum turning rate of the UAV is 2 degrees per second, which means that if the UAV wants to fly a circle above a certain object, this takes 3 minutes or 6 pictures. Flying a circle above a target also means that the end of the search path is shortened by 3 minutes or 6 pictures.
3.2.2 Solution Approach

We evaluate search paths by means of an objective function, based on (expected) value functions. This evaluation is needed for 1) the a priori calculations for determining optimal search paths, as well as for 2) in-flight adaptation of a search path. For the former (a priori search process), we provide more details in the following section. For the latter (in-flight adaptation), we provide details in this section after explaining the used value functions. We employ two different functions for evaluation: first, the value function, that computes the total value of a path after flying; and second, the expected value, that estimates the value of a (partial) path before flying and, in case of the adaptive agent, during the flight.

The value function is defined as:

\[
V = \sum_{t=1}^{T} \sum_{n=1}^{N} \text{utilityGain}(n, t)
\]  

where \( T \) is the number of discrete time intervals during the mission, \( N \) is the number of detected targets at time \( t \) and \( \text{utilityGain}(n, t) \) is the gain in utility of information for target \( n \) at time \( t \).

The utility gain function \( \text{utilityGain}(n, t) \) can be interpreted as the number of points scored for observing a target. Upon first observation of a target, the utility gain is 1. This increases linearly with time for the duration of observation of this target with a maximum utility gain of 6 per target. The reason for this maximum is that identification cannot improve after 6 detections. However, after 6 consecutive non-detections (when a target seen before is now undetected), known information about that target is reset which means that when the UAV encounters that target after that time, new information can be gained yet again for that target.

We define the expected value function of a UAV search path as:

\[
E(V) = \sum_{t=1}^{T} \sum_{c=1}^{C} p_{\text{det}}(c) p_{\text{target}}(c)
\]  

where \( T \) is the number of discrete time intervals during the mission, \( C \) is the number of cells within the detection range of the UAV at time \( t \), \( p_{\text{det}}(c) \) is the probability of detection. This number depends on the type of terrain at cell \( c \) and \( p_{\text{target}}(c) \) is the probability of a target being present at cell \( c \). We assume this
information to be available and, because of the high resolution of the terrain, we also assume that no more than one target can be present at each cell.

This formula thus estimates the number of targets that will be detected during the length of the mission based on the probabilities of 1) the presence of a target and 2) detection by the UAV.

### 3.2.3 UAV adaptive agent

The UAV agent determines the behaviour of the UAV in terms of adapting the flight path or not. The *online adaptive agent* will decide on flying a circle above a detected target based on the expected value of the remaining search path. Pseudocode for this agent is depicted in Algorithm 3.1, that runs each timestep of the flight, when a picture has been taken.

/* The UAV starts flying the predetermined search path. At each timestep when a picture is taken and analysed, the following code is executed: */
if the UAV detects a target that has not been seen before then

/* Determine the expected value of the rest of the search path (from current timestep \( t \) until the final timestep \( T \)) */
expValueWithout = \( E(V)_{t,T} \);

/* Determine the expected value of the search path when a circle is made. To do this, the expected value gets 6 points for the circle (unless the expected value of the circle is greater than 6) and the rest of the path has been made 6 steps shorter. */
expValueWith = \( E(V)_{t+6,T} + \max(6, E(V)_{t,t+5}) \);

if expValueWith > expValueWithout then
  flyCircle();
else
  keepFollowingOriginalPath();

**Algorithm 3.1**: The algorithm for the online adaptive UAV agent.

When the UAV is currently not flying a circle (because otherwise the UAV could start flying circles within circles and this would increase the complexity of getting back on the original path significantly) and a new target has been observed, two values are computed: the expected value of the rest of the search path without flying a circle (expValueWithout) and the expected value of the rest of the search path with the certainty of observing a certain target during the circle (expValueWith).
3.3 Experiments

In this section, we describe the experimental design and setup, the results we obtained and an analysis of these results.

3.3.1 Design and Setup

The main objective of this research is to investigate if our online adaptive UAV agent improves the value of a predefined search path. To this end, we compare our agent, as described in section 3.2, to two benchmark agents: The Naive Agent, in which the UAV has a predefined search path and the UAV will just follow this path without doing anything differently. The Exhaustive Agent is the other benchmark and has predefined online behaviour: the UAV starts flying the predefined search path and each time the UAV detects a target, it always decides to fly a circle around that target before continuing its path. This agent is necessary in our experiments, because if we want to show that it is beneficial for the value to sometimes fly a circle, we also need to show that it is not a good idea to always fly a circle.

Our experimental design has 3 independent variables that we systematically vary to investigate the effects: 1) target distribution, 2) search path and 3) agent type.

- Target distributions: We have generated 10 different target distributions, each consisting of 1,000 targets, placed in the terrain using the distribution as shown in table 3.1(b). For each type of terrain, the targets are normally distributed.

- Search paths: We run the experiments on 10 different search paths. We generated search paths by hand and we ran a simple Particle Swarm Optimisation (PSO) technique (Eberhart and Kennedy, 1995) to optimise these search paths based on their expected value. This work closely resembles the work described in Pitre et al. (2009). After we ran the PSO algorithm for a fixed amount of time, we picked the 10 best paths for use in our experiments.

- Type of agent: As explained above, there are three types of agents: the naive agent (without any online adaptation), the exhaustive agent (that will always fly a circle upon detection of a new target) and the adaptive online agent (that will base its decision of flying a circle on expected value calculations).
Chapter 3. Autonomous UAV Path Planning

Figure 3.2 – (a) shows an example naive path, without online adaption; (b) shows an example exhaustive path, with many circles during the flight; and (c) shows an example adaptive path, with some circles here and there.

The main measurable is the obtained value of a search path given a type of agent. The higher the value of a search path, the better. For each combination of a search path and a target distribution, we measure the value of the paths that are generated by the three different agents. We hypothesise that the utilities of the paths generated by the adaptive agents are better than the utilities of the paths generated by the naive and the exhaustive agents. We also measure the number of detected targets and the total number of detections. Using these two metrics, we can see to what extent the different agents are better in searching, identification or both.

The different types of terrain and the detection probabilities of the different types of terrain were explained above in Section 3.2. The UAV starts flying in the bottom right corner of the world.

3.3.2 Results

Before we present the results of our simulations, we give some illustrative screenshots of the simulation, showing different kinds of search paths (albeit somewhat simplified for reasons of clarity). Here, the UAV starts in the bottom right corner of the terrain and each green dot is a location at which the UAV takes a picture which is then analysed using one the three agents. An example flight is shown in Figure 3.2. In Figures 3.3(a) and 3.3(b), the results for every run are shown in terms of value differences between the adaptive and the naive/exhaustive agents, respectively. On the x-axis of these charts are the 10 different target distributions. For all these 10 target distributes, the results for the 10 different search paths that
3.3. Experiments

Figure 3.3 – Differences between the adaptive agent and the benchmark agents.

We used are shown. On the $y$-axis, the difference in value is shown. Figures 3.4(a) and 3.4(b) are two histograms of the data from Figures 3.3(a) and 3.3(b). From these histograms, it becomes clear that the data is not normally distributed, but slightly positively skewed. In the next section, we analyse this skewness. We also have included an example graph of this in Figure 3.5. The figure shows for each timestep the value of the search path up until that point. All lines are non-descending, since value will only increase over time. In Table 3.2, the mean values for the total number of detections per run of the different agents is shown, as well as the mean number of uniquely detected targets per run. The ratio between these two values, which gives an indication on how well the identification objective is executed, is also included in this table.

3.3.3 Analysis

From Figures 3.3(a) and 3.3(b), we can see that the adaptive agent generally performs better than the naive method and much better than the exhaustive method. Some exceptions occur, for instance distribution 7. We analysed these exceptions and these UAV paths do not encounter as many targets as expected.

The difference between the exhaustive and the adaptive agent are much larger. When many circles are flown in a short period of time, many targets will be detected for many more than 6 times, which yields no further utility gain. The histograms in Figure 3.4 are positively skewed. Using the Wilcoxon Signed-Rank test, we found that the adaptive agent is significantly better than the naive and exhaus-
Table 3.2 – The mean values for the number of uniquely detected targets, the total number of detections and the ratio between these values.

<table>
<thead>
<tr>
<th></th>
<th>Naive</th>
<th>Exhaustive</th>
<th>Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td># targets</td>
<td>171.04</td>
<td>68.44</td>
<td>148.99</td>
</tr>
<tr>
<td># detections</td>
<td>315.04</td>
<td>362.22</td>
<td>347.09</td>
</tr>
<tr>
<td>detections / targets</td>
<td>1.84</td>
<td>5.29</td>
<td>2.33</td>
</tr>
</tbody>
</table>

tive agents using a significance level of \( p = 0.05 \), which validates our hypothesis. Figure 3.5 depicts an example run. In this Figure, we observe that the naive agent does not significantly differ from the expected value. The exhaustive agent starts out well, but is outperformed by the other agents after some time. Note that Figure 3.5 is an example of one single run. Plots of other runs look differently. This can also be derived from the other plots; sometimes the naive or exhaustive agents are better. But generally, the plots follow this pattern.

Our second metric, i.e., the number of detections versus the number of uniquely detected targets, is depicted in Table 3.2. Using the numbers from this table, we can say something about strengths and weaknesses of each agent. We expected the naive agent to be the best in searching, the exhaustive agent to be the best in identification and the adaptive agent to be the best in jointly optimising these objectives. The naive agent has the highest mean number of uniquely detected targets, while the exhaustive agent has the highest ratio between the number of detections and the number of targets. The adaptive agent is best in jointly optimising these objectives.

Figure 3.4 – Histograms of the differences between the adaptive agent and the benchmark agents.
3.4 Conclusions

In this section, we propose a UAV agent that on-line adapts its predefined search path according to actual observations during the mission. The adaptive agent flies a circle above a detected target when it expects that this will improve the total value of the search path.

Our results show that our agent significantly outperforms both a naive and an exhaustive agent. However, not in every instance the adaptive agent outperforms the other two; in some cases one of the benchmarks is better. This result can be attributed to unexpected situations during the flight.

We also conclude that each agent has its own strength. It depends on the user’s goal which agent is best. In our scenario, we want to jointly optimise search and identification objectives. Using these objectives jointly, our adaptive agent outperforms the benchmarks. But if searching was the only objective, the naive agent would be better; likewise, when identification was the only objective, the exhaustive agent would be the better one.

As a future research path, we will generalise the model further by introducing different kinds of vehicles with different kinds of capabilities (e.g., helicopters, ground vehicles, underwater vehicles). We will investigate how to model different capabilities and how the different vehicles in the field can make use of other vehicle’s capabilities. Related work in this direction has been done by Kester (2008) to find a unifying way of designing *Networked Adaptive Interactive Hybrid Systems*.
Part II

Autonomous Driving
Introduction

Rapid developments in the ICT-domain will have a huge impact on future transportation systems. These developments come with a number of issues. Firstly, projections about the future usually focus on what is technologically feasible without paying proper attention to what the actual needs are from the personal perspective (the perspective of the vehicle owners, drivers and travellers) on the one hand and needs from the public perspective (the perspective of the traffic managers, communities and governments) on the other hand. Secondly, projections are made separately for different technologies like communication, information fusion, control or artificial intelligence, not in combination. A third issue is that the roadmap that reaches the projected ideal situation is not always clear, as each consecutive step on the roadmap should have added value as well. In this introduction to part II of this thesis we would like to address these issues.
The approach taken in here is that we first identify important technological developments in the ICT-domain for transportation systems. Then, we analyse what the needs are from a personal perspective (the perspective of the vehicle owners, drivers and travellers) and from the public perspective (the perspective of the traffic managers, communities and governments). We then evaluate what technological innovations meet the needs best, as well as the costs and the feasibility of these technological innovations. After we discuss non-technical issues, we discuss the possibilities of getting from the current situation to the projected future transportation system that will meet the personal and public needs best.

**Technology Development**

Technology in the ICT domain is developing rapidly and is expected to do so at a similar pace in the next decades. Those developments lead to various transformations important for the transportation systems domain:

**From vehicle to smart transporter**  Vehicles are getting more and more equipped with many different sensors, such as radars, lasers, gyroscopes, GPS units, cameras and acoustic sensors. Powerful algorithms with more and more processing power process and fuse all the information to create situation awareness, including self-awareness. Based on this awareness many different capabilities can be built, such as automatic obstacle avoidance, adaptive cruise control and lane keeping. Several vehicle manufacturers and companies like Google are even developing fully autonomous driving. These developments transform the vehicles into personal (or public) ‘smart’ transporters.

**From mobile phones to personal smart companions**  Similarly to cars, mobile phones are also equipped with more and more powerful sensors. Furthermore, the evermore powerful computing power makes it possible to run many applications on these devices turning the mobile phone into a personal ‘smart’ companion. Those companions can take different forms, like vests, helmets or glasses.

**From roads to supportive smart infrastructure**  With all these fast developments the (road) infrastructure seems to be lagging behind. The main reason for slow innovation is that investments are financed by public money, which means that
a lot of parties are involved and public interest must be covered a priori. There are however interesting developments in web-enabled devices, similar to smart companions, that can accelerate developments considerably. When these devices become sufficiently cheap and small, they could be integrated with the lighting infrastructure at low additional cost, resulting in a smart and supportive infrastructure.

From communication systems to communication services  A lot of attention will go to the interaction between smart vehicles, smart personal companions and smart infrastructure. Development of these systems, therefore, relies on developments of communication services available to them. The most important means of communication for those devices will be Wifi, 3G/4G and infrared. The trend here is that the communication systems are more and more seen as entities trying to maximise a communication service to all devices in the cloud. Such communication systems therefore also become smart and could for example make use of cognitive radio techniques (Mitola and Maguire, 1999) in order to use the available spectrum in the most efficient way. We could also refer to such integrated developments as the internet of (smart) things in the intelligent transportation systems domain.

Projections about Future Transportation Systems

From such a technological analysis, predictions about the far future often turn out to be too pessimistic on the rate of innovation. The main reason for this is that inevitably, predictions have to be made from current knowledge. Unexpected discoveries or accelerated developments are hard to take into account.

On the other hand, predictions about the near future turn out to be too optimistic on the rate of innovation. This is because most effort is spent on what is technologically feasible while there should be more focus on whether there is a real need for it and if one is willing to pay the price. Furthermore, there are psychological and social issues that can influence the course of development. Finally, there is too much focus on the end result. It will only become a reality if there is a suitable roadmap, meaning that every step in the development has to be beneficial in itself as well.
Therefore, in order to get a better grounding for the developments for transportation systems, we will address a couple of issues. First, we address what is needed from a personal and a public perspective. Then, we discuss technological solutions that meet these needs. We also discuss the costs, psychological, social and political factors, and some possible roadmaps to meet these needs.

**Personal and Public Needs**

From a personal perspective, we identify a couple of needs that future transportation systems need to adhere to. Safety is the first important issue – if these systems would not be safe, they would not be adopted. The minimisation of travel time is a second need. Third, such systems must be affordable for the masses. Fourth, a certain level of comfort should be guaranteed. Fifth, intelligent transport systems should not increase the cognitive load for humans in the vehicles. Sixth, these systems should keep the driver informed about the situation on the road. As a seventh and final personal need, these systems should preferably be environmentally friendly.

Different drivers may have substantially different needs. Therefore, an additional need can be identified: the ability of the car to cope with different needs from different drivers and even with changing needs of the same driver. This need is the main motivation behind the research that is described in chapter 5.

From the public perspective, safety is also the first important need for future transportation systems. Second, these systems should preferably be beneficial for the economy. The third and final need we identify from the public perspective, is that these future systems should be environmentally friendly.

These public and private needs are largely drawn from common sense, although many of them also appear in the roadmap from the Dutch Ministry of Infrastructure and the Environment (Ministerie van Infrastructuur en Milieu, 2013).

**Meeting the Needs**

How can we meet these needs? If we consider the current situation, we appreciate that the technological developments discussed above could meet needs on safety, comfort, lower cognitive load and being well informed. Environmental friendliness depends much more on whether traffic will be less polluting than it is now. This
could be achieved by a substantial increase of the number of electric cars, which depends highly on better and cheaper battery technology. Recent developments in that field are very promising. For example, the use of graphene in lithium-ion batteries has proven to improve the battery life substantially (Lin et al., 2013).

This leaves us still with the most important needs: short travel time, low travel cost and economic benefit. It is clear that this has to be solved by somehow making the traffic more efficient, i.e. getting more cars from $a$ to $b$ without increasing cost proportionally. When we consider the wide variety of applications that are proposed, such as improved traffic light control, ramp metering, assisted merging, adaptive cruise control, shockwave damping and traffic network management, they typically can improve efficiency in the order of 5-20% (see for example Van Arem et al. (2006)).

There is, however, one application, cooperative adaptive cruise control (CACC), that has the potential of improving the efficiency substantially. In CACC, the cars communicate, possibly aided by the road side, their kinematic information. This results in much shorter possible headway times between vehicles, effectively creating trains of vehicles, or platoons (see Bergenhem et al. (2010)). In table 3.3, the throughput of the traffic on a highway is shown for cars driving at different velocities and different headway times. The values in this table are computed by the following formula:

$$\text{throughput} = \frac{\text{distance travelled per hour}}{\text{vehicle size} + \text{velocity} \times \text{headway time}}$$  \hfill (3.4)

where speed is measured in m/s and throughput in number of vehicles per hour. We assume a vehicle size of 6 meters and the values in the table are rounded off. When headway times are large, the throughput is low and not much dependent on the velocity, while at shorter headway times, the throughput increases dramatically, particularly at larger velocities. In the current situation, the throughput on highways is typically around 2000 vehicles per hour. When headway times are decreased, the throughput can be increased by 200-400%.

In table 3.4, the throughput of cars is given in a typical urban scenario with traffic lights. The values in this table are computed by the same equation as described above, with the added assumption of a traffic light that has a green light for 20 seconds. An extra variable in this scenario is the vehicle acceleration, because the vehicles need to start moving when the traffic light turns green. This acceleration is dependent on the desired velocity of the vehicle.
In such urban scenarios, an increase in throughput of more than 200% can be achieved as well when headway times are substantially decreased.

If vehicles could also steer autonomously, needs could be met even further. Drivers could attend to other things while driving. Furthermore, they could drive to the centre of a city, get out of the car and the vehicle would park itself at the outskirts of the city. Such vehicles could just as well be used for public transportation, bringing drivers from the place they are to the place they want to be without the need for additional infrastructure.

This may sound a little futuristic, so in the next section we will look at the technological feasibility of such systems.

### Feasibility

Vehicle manufacturers and companies like Google are focusing on technologies for individual vehicles, such as collision avoidance, adaptive cruise control or even fully autonomous driving.
However, to realise high throughputs, communication between vehicles at a high frequency (typically in the order of 10 Hz) with low latency is necessary (Eckhoff et al., 2013).

Current implementations of CACC focus on the control algorithms where the boundary conditions are safety (they don’t collide) and string stability (speed fluctuations decrease as they propagate along the vehicle stream). Best results are achieved when the intended or desired acceleration of the preceding vehicle is known, because it predicts the behaviour of the preceding vehicle better.

While this seems straightforward, a number of technological problems occur. The estimates of own state and states of the preceding vehicle(s) have uncertainties because of imprecise sensor measurements. Furthermore, the vehicle behaviour model of the preceding vehicle can be uncertain and communication between the vehicles may fail occasionally.

This means that CACC implementations are needed that can deal with these uncertainties, including complete failure of sensors and communication. In order to do that, the CACC algorithms have to actively reason about the effect of uncertainties on safety and behave in such a way that safety is guaranteed at acceptable levels under all circumstances.

Recent developments have shown that acceptable levels of safety can be guaranteed if the control system becomes more cognitive of and adaptive to the particular situation. The control algorithms adapt their headway time depending on the uncertainty of their own state, the state of the preceding vehicle(s), the expected quality of the communication service (packet loss and latency) and the uncertainty of the vehicle model. In this way, CACC can result in significantly higher throughput at an acceptable safety level. In chapter 4, we tackle this issue in more detail.

When it is known how the performance of the CACC algorithms is affected by the quality of the information, it is possible to formulate, quantify and reason about how much their performance would improve if they had more and/or more accurate information or, in other words, how utile this information would be.

Another important innovation can now be realised. When we can reason about how utile information is to realise the system goals, it is also possible to reason about the utility of information exchange between the various information sources. We can explicitly reason about how utile it would be to send out a request to other information services or, in the case of time based communication, adjust the power and/or communication frequency such that system goals are met best.
This can be extended to optimal use of all communication services, like Wifi, 3G/4G and infrared, by reasoning how much the expected performance of the system would improve when using a certain communication service or adjusting the communication service to the current situation.

Important to note is that the systems goals may change on the fly, for example because the driver wants more comfort or extra safety or the internal utility function for information changes due to changing situations. The system therefore needs to and can, adapt itself constantly.

The expectation is that these technology developments will bring the headway times well below 0.5 seconds at acceptable levels of safety thereby improving throughput by more than a factor of 200%. We also confirm this in chapter 4.

If all vehicles would have advanced CACC capabilities, automatic steering would also be easier to accomplish because the trajectory of the predecessor is known and automatic steering would then be equivalent to automatic following of this trajectory.

The systematic approach as described above is most crucial for CACC but could and should also be applied to other control applications at a larger time scale, such as ramp metering, lane merging or shifting, shock wave damping and network traffic management. This may even be more important in a situation where only a fraction of the vehicles is fully equipped. We conclude that if a systematic integrated approach is taken, as described above, a traffic system that meets the needs is technologically feasible.

Cost

What would be the cost involved to realise such a system? We consider a number of cost factors.

Cost of vehicle equipment The equipment needed for intelligent vehicles is GPS, radar or laser, cameras, vehicle motion meters, gyroscope, a wireless communication system and a computation unit. None of these components are very expensive. It is expected that when the CACC systems are made in substantial numbers the whole systems can be made for a fraction of the price of the vehicle itself.
Cost of vehicle insurance  Insurance companies look at total profit. If the cars are safer, which is the strongest requirement, they will have no trouble insuring it. If there are only a few equipped cars it has no significant effect on their profit so they even allow cars that are less safe (e.g. ACC) than the cars proposed here.

Nowadays, there is already quite a number of vehicles on the road with adaptive cruise control (ACC) which is less safe than CACC, yet there are no problems getting insurances for such vehicles. Insurance companies only look at the big number and will adjust their policy when it becomes too costly. As explained above, safety is one of the mean issues that can be actively guarded with the proper technology.

Cost of roadside equipment  The technology needed to realise a smart infrastructure with web-enabled sensors will heavily build on the technology for smart companions. Since this will be massively produced, the additional cost to the current road-side lighting infrastructure is expected to be low.

Cost of personal smart companion  The development of smart personal companions is still moving rapidly. It is driven by the consumer market and in the future, almost everybody will have a smart companion with ever-increasing capabilities, irrespective of their need for transport. Therefore, apart from improvement in the interaction with the smart transportation systems, no additional costs are expected.

Cost of development  The steps that are contained in the roadmap are decisive for the eventual cost of research and development. We will therefore address this issue in the section where we consider the possible ways to move forward.

Psychological and social issues

We identify a number of psychological and social issues, that may have negative impact on the development of intelligent transportation systems.

Drivers may not be inclined to drive at the shortest possible safe headway time. Studies have shown that both male and female drivers tend to drive at short, if not the shortest possible headway time that is considered safe (Shladover et al., 2013; Nowakowski et al., 2010). This is partly due to the safer feeling CACC provides
compared to ACC. This is because the responsive behaviour of CACC is also much better.

**Drivers may not be interested in driving closely behind a predecessor and therefore will not buy such a vehicle.** Indeed, there seems to be no direct personal interest to buy such a vehicle. We address this issue in the next section.

**Drivers may not like to be controlled by the traffic manager or an automated road side.** Drivers generally like their freedom and are may not be very enthusiastic about the prospect of buying an equipped car and being ‘controlled’ by a smart road or traffic manager. This is the main reason why the road side should also provide useful services to the driver, such as information about the traffic situation ahead or speed advice. It is up to the driver to make use of the services provided.

**Drivers may not like the fixed dependence on the behaviour of their predecessor when driving in a platoon.** Predecessors will also be equipped and probably use policies that optimise on comfort to drive through traffic. However, if they prefer a faster, more aggressive driving style, these preferences can be communicated and drivers are free to join other platoons that suit their preferences better.

**Drivers may be sceptic about safety in autonomous driving.** In four of the United States, autonomous vehicles are already allowed on the road. Of course, malfunctioning autonomous vehicles will get a lot of attention. However, when the overall safety of vehicles still increases, there is no reason to believe such systems will not be legalised.

**Liability is unclear. When is the driver responsible and when the car manufacturer?** Liability issues will probably not be much different from the current practice. For improper use the driver is responsible, for malfunctioning of the vehicle the manufacturer is responsible.

**It has always been difficult for scientific and technological communities to cooperate.** As explained before, cooperative development of communication, information and control domain is needed, while keeping the developments in cogni-
tive and intelligent systems into account as well. However, each scientific domain is typically focused only on developments that are considered important within their peer group. These groups usually have different perspectives about the architecture of the transportation system and the crucial technologies that need to be developed. This hampers an integrated cognitive system approach and may confuse system engineers. Fortunately, complex adaptive and intelligent systems engineering is gaining weight as a scientific field in itself and focuses explicitly on being able to cope with these differences of opinion in a more systematic and even automated way.

**It is hard to create public awareness of new technology.** Although some capabilities are already available, like adaptive cruise control, most people are not aware of it. Yet once they have had some experience with it, they tend to stick with it (Shladover et al., 2010). Getting the benefits of new technology under the attention of the public is a good investment.

**People will be tempted to disrupt the system.** Any system that can be made to fail with considerable consequences attracts malicious people. However, we argue that the possibilities for these people are limited because, as described above, the system is made such that it understands the situation and can act to maximise utility. For example, if the communication is jammed, the vehicles will automatically adapt the control such that it stays safe. Deliberately false information can be double checked with other information sources for consistency. Furthermore, outsiders cannot have access to critical systems because each sub-system interacts with other sub-systems on the basis of service requests and service provision. Finally, people trying to disrupt the system can be spotted immediately by the smart systems on and beside the road.

**Roadmaps**

How can we get from the current situation to a situation in which vehicles are fully equipped with advanced sensors and adaptive algorithms, seamlessly integrated with smart personal companions and supported by an adaptive information infrastructure? Although the benefits of such a transportation system are huge, to arrive
from the current situation to that situation in a cost efficient way is a different matter altogether.

Various roadmaps are already being implemented. The United Nations have created a roadmap (UNECE, 2013) that focuses mainly on international cooperation on the short term. This roadmap contains 20 actions to move closer towards intelligent transportation systems. In the Netherlands, the Ministry of Infrastructure and the Environment have also created a roadmap for ITS (Ministerie van Infrastructuur en Milieu, 2013). This roadmap defines 6 transitional paths towards ITS that should be implemented by the year 2023.

In Hoogendoorn-Lanser et al. (2011) and Hoogendoorn et al. (2011), multiple future visions are described, considering several stakeholders, and the implications of these future scenarios.

Certain developments in enabling technologies will steadily continue without much involvement from public funding. Car manufacturers will develop technologies that will naturally appeal to vehicle owners. For technology that is mainly beneficial for the community, extra measures have to be considered that will speed up its introduction and acceptance.

More facilities on the road side will become available, such as ramp metering, shock wave damping and traffic light control. But how do we speed up the use of CACC or platooning capabilities specifically? The roadmaps mentioned above focus on the general development of intelligent transportation systems. We focus on the adaptation of CACC specifically, and identify a couple of relevant measures.

Promote new technologies that is in the public interest Certain technologies would be adopted sooner if people were aware of their existence and if they had some experience with it. A good example of this is ACC. Governments should actively promote such technologies that are in the public interest, either by advertisement, subsidy or technology development. The criterion should be that expected gains are higher that the promotion cost.

Trucks with CACC or platooning capabilities A natural application for CACC or platooning is freight transport. This could increase safety, decrease fuel consumption and finally also decrease the number of truck drivers needed. Introduction would be easier since the trucks could at first be of the same company, which enables a more homogeneous and thus less complex implementation.
Sponsored leading vehicles  Sponsor people, e.g. through tax cuts, who are willing to act as leading vehicles in platoons. Another option is that they get rewarded for each vehicle they are able to lead through the traffic.

Supportive infrastructure  Extra information could be provided by the infrastructure to equipped vehicles about non-equipped vehicles. Based on this information, equipped vehicles could better decide how to behave in the best interest of the driver, while driving through traffic consisting of a mixture of equipped and non-equipped vehicles.

Equipped platooning vehicles are granted extra communication and other services  An interesting approach to interest people to buy and use CACC or platooning equipped vehicles is to grant extra free services if vehicles engage in CACC or platooning. These extra services could include an internet connection and other information / entertainment services.

Which of these various ways to achieve this will turn out to be most cost effective is currently hard to predict. However, that this difficult hurdle will be taken one way or the other is plausible, as there are many benefits.

Part II

This part of the thesis consists of two chapters, that each tackles a very specific subject within the domain of ITS. In chapter 4, we describe our research into the computation of a minimal safe headway time, given uncertainty of various sensors and system parameters.

In chapter 5, we describe a somewhat more futuristic scenario, in which we try to evolve controllers that each work according to a different prioritisation of a set of objective functions. The idea behind this is that drivers on the road may have different needs, and want their vehicle to drive according to a user-specified set of objectives, such as speed, comfort or fuel economy.
Safety in the Face of Uncertainty

The introduction to this part of the thesis detailed several challenges for the development of intelligent transportation systems. One of the challenges is how these systems remain safe. When intelligent systems enable autonomous vehicles on the road to drive closely to each other, safety is a critical issue.

The sensors, communication modules and system behaviour are inherently uncertain. Sensors can fail or give erroneous readings, communication links can drop or be delayed and the vehicles themselves may also react differently from expectations.

This chapter proposes a solution to this problem. If the uncertainties and delays are known while driving on the road, it turns out that it is possible to reliably approximate the critical safe headway time that vehicles have to minimally adhere to.

The sections in this chapter show how we iteratively tackled this problem. Each section builds on the lessons we learned from the one before. Section 4.1 is the result of our first effort. In this section, we modelled different adaptive cruise controllers and simulated how many times two consecutive vehicles would crash in an emergency stopping situation, given some factors of uncertainty in the system.
and given a certain velocity and inter-vehicle distance. The result of this study is a
look-up table of minimum safe headway times in different scenarios.

The main insight that triggered the work that is described in section 4.2 is that
the difference in displacement of two vehicles is a much better metric to calculate
the minimum safe headway time. In this section, we introduce the notion of base
distributions, that give an indication of what the difference in displacement between
two vehicles would be, given uncertainty in the circumstances and given several
models of (cooperative) adaptive cruise control. Instead of counting the number
of crashes given a certain setting, we would derive the minimum safe headway
distance from these base distributions.

The greatest disadvantage of the approaches mentioned in sections 4.1 and 4.2,
is that they are computationally very expensive, due to the Monte Carlo simula-
tions that we ran in those experiments. In the final section of this chapter, section
4.3, we provide a solution for this problem. This section describes an analytical
model for computing the critical headway time. The great advantage of this ap-
proach is that this model is very fast.

We validated the analytical model with the earlier mentioned Monte Carlo sim-
ulations and the results are very promising. One of the assumptions under which
this model works is that the uncertain parameters are normally distributed. How-
ever, we also show that even when some of the parameters aren’t normally dis-
tributed (which is for example the case with the communication delay model), the
increased computational cost is manageable.
4.1 Evaluating Adaptive Cruise Control Strategies in Worst-Case Scenarios

This section is concerned with safety in (cooperative) adaptive cruise control systems. In these systems, the speed of the cars is maintained automatically, based on the preferred speed of the driver and the speed of the preceding car. Technologies that are used in these systems, such as radar and radio communication, introduce many factors of uncertainty in the system. In this section, we present models for different adaptive cruise control strategies, in which this uncertainty is explicitly modelled. By simulating emergency braking situations under these uncertain circumstances, we find the minimal safe headway time for these strategies.

4.1.1 Introduction

Nowadays, many commercially available cars have adaptive cruise control (ACC) functionality. This extension to normal cruise control uses a radar to determine the distance and relative speed of the preceding vehicle and controls the acceleration based on this information. This functionality increases comfort and safety of the driver.

However, traffic throughput is not necessarily improved: ACC-equipped vehicles still maintain a rather large following distance, which has two reasons: first, it is a convenience system, not a safety system, so the human driver should have enough time to correct errors made by the ACC, should they occur. Second, this distance is needed in order not to exhibit aggressive driving behaviour. Traffic throughput would be improved if cars would be able to drive closer to each other. To this end, the notion of cooperative adaptive cruise control (CACC) is introduced. This extension to ACC functionality includes direct radio communication

Section 4.1 was published as:

between vehicles. This enables a car to directly communicate its change in acceleration to its predecessor, which leads to faster response times.

Current ACC-equipped vehicles maintain a headway time of at least 1 second to their preceding vehicle, which is about the same headway time as humans keep on highways. This distance is ‘safe’, in the sense that if a preceding car does an emergency brake, there is enough time for the radar and/or the driver in the following vehicle to react to this. However, in CACC-controlled vehicles, the headway time (in the order of 0.1 seconds) is much too short for humans to react to emergency brakes. This means that drivers in CACC-equipped vehicles have to rely solely on the car’s ability to detect emergency brakes and to react accordingly. Determining the minimal safe headway time is therefore essential for CACC-equipped vehicles.

In this section, we experimentally determine the minimal safe headway time for three different controllers: the ACC controller, that uses radar technology to derive information about its preceding vehicle; the CACC1 controller, that communicates the value of the car’s accelerometer to the following vehicle; the CACC2 controller, that has a built-in braking model, that estimates the change in acceleration directly after a braking action occurs, which is before the car starts decelerating. We test these controllers using different initial velocities and different initial distances between the cars.

In our models, different factors of uncertainty are introduced, such as uncertainty in accuracy of radar readings and uncertainty in communication success. The values for these uncertainties are based on realistic values that occur in currently used technology. This uncertainty makes this model realistic and therefore useful in practice.

Design of CACC is in general concerned with two main issues: safety and so-called string stability (Ploeg, 2014). Most current research is about the string stability issue, but the safety issue is largely ignored, especially when taken into account the uncertainty in information and communication that these systems have to deal with.

In this section, we only focus on the safety aspect. Our safety controller should be able to react to an emergency brake by its preceding vehicle when necessary. The comfort controller is concerned with keeping a platoon of cars smooth (i.e. maintaining a steady velocity) and string stable. String stability in a platoon means that oscillations in speed within the platoon will be damped by the following vehicles instead of amplified.
4.1. Evaluating Adaptive Cruise Control Strategies in Worst-Case Scenarios

4.1.2 Related Work

In intelligent transportation systems (ITS), we can identify different branches of research: one is focused on developing intelligent vehicles in order to improve throughput in highway traffic and the other is focused on including intelligence in road infrastructure to improve this. The different branches enable different applications. Smart infrastructure can enable traffic monitoring and is also able to improve traffic throughput by means of giving speed advice to the drivers or directly to the adaptive cruise control.

Our interest lies in the development of intelligent vehicles. A first step into this direction is the development of adaptive cruise control, in which a car computes its distance to the preceding vehicle by means of a radar. Marsden et al. (2001) provide a comprehensive article about this technology and its implications on traffic flow.

Taking this technology a step further, we come in the domain of cooperative adaptive cruise control (CACC), in which direct communication between vehicles allows them to react faster to each other. Van Arem et al. (2006) describe the effect of CACC on traffic flow. They conclude that, when the penetration level of CACC-equipped vehicles is high enough (> 60%), traffic stability and throughput is improved. Yang et al. (2004), a communication protocol is proposed in order to make a cooperative collision warning system on highways.

The main application area of CACC technology these days is platooning. Broggi et al. (2000) and Kanellakopoulos et al. (1999) both use image recognition techniques in combination with sensors to autonomously enable platooning. However, current technology has improved significantly since then and nowadays direct radio communication between vehicles is used to enable platooning.

In Hallé (2005), an extensive architecture is given for a layered multi-agent CACC architecture. The authors use this architecture to implement both centralised platoons (in which there is a coordinating platoon leader) and decentralised platoons (in which all cars operate as equals). Khan et al. (2008) present different platoon (in their paper, convoy) forming strategies, based on a utility value of a platoon.

One of the missing elements of the above approaches to designing CACC systems, is that they do not explicitly account for uncertainty in information and communication. Machine learning techniques could be a promising new way of autonomously learning the uncertainty in the vehicles.
An important aspect of platooning is ensuring string stability within a platoon. A thorough control-theoretic model of string stability in CACC is presented by Naus et al. (2010).

To conclude, current research is mainly about CACC and how to design and implement these systems. We found that the safety aspect of the problem, while taking uncertainty of information and communication into account, is not often considered, although it crucial to the eventual acceptance of such systems by the public. Therefore, this will be the main focus of this research.

### 4.1.3 Model

In this subsection, we describe the model that we used in our simulations. Since there are many sources for uncertainty in the model, a mathematical analysis of our model is complex. We also intend to use this model as a starting point for more experiments, that would include more complex behaviours and more uncertainty. This well justifies our choice for simulation.

**Cars**

We model three types of car: cars containing an adaptive cruise control (ACC) controller and two different types of cooperative adaptive cruise control (CACC\(_1\) and CACC\(_2\)) controller. The main difference between the ACC controller and the two CACC controllers is that the ACC controller uses radar technology to derive information about the preceding vehicle, whereas the CACC controllers use direct communication to derive the same information. This difference in technology has some implications. First, radar can only derive information about distance and (relative) velocity. This means that information about the acceleration of a preceding car has to be derived from this information. Radio communication can be much more informative, because any information can be transmitted, if the car considers it useful. In our setting, cars only communicate their acceleration. Second, information that cars know about themselves is generally more accurate than information that is derived by a radar.

The controllers only control the acceleration of the vehicle. Since our experiments take place on a highway with only 1 straight lane, there is no need for incorporating steering or lane changing in our model.
There are properties that apply to both the simulation of the ACC and the two CACC controllers. They both share the same update scheme, that we use in our discrete time-based simulation. This scheme is depicted in Algorithm 4.1.

\[ \Delta t = 0.01 \]

```
foreach timestep t do
    \( \ddot{x}_t \leftarrow \text{compute new } \ddot{x}; \)
    \( \dot{x}_t \leftarrow \dot{x}_{t-1} + \ddot{x}_t \Delta t; \)
    \( x_t \leftarrow x_{t-1} + \dot{x}_t; \)
```

**Algorithm 4.1:** Updating scheme for cars

In this scheme, \( \ddot{x}_t \) is the acceleration of the car at time \( t \), \( \dot{x}_t \) is the velocity of the car at time \( t \) and \( x_t \) is the position of the car on the road at time \( t \).

There is a small delay between a car’s braking activity (i.e. pressing the brake pedal) and the actual deceleration of the car. This delay is 150ms. This means that when a car decelerates at \( t = 0 \) and the car behind hits the brakes at \( t = 0.5 \), the car will start decelerating at \( t = 0.65 \).

In this model, cars are not allowed to brake harder than \(-9\text{m/s}^2\) and they will always obey this law. The maximum deceleration for each car is \(-9\text{m/s}^2\), but sometimes cars brake less hard than they think they do, due to mechanical limitations. See the subsection on uncertainty below for more details about this.

The difference between the ACC and the two CACC controllers lies in the updating rule for the acceleration. In the following subsections, these updating rules are described in detail.

**ACC controller**  The ACC controller uses radar technology to detect the acceleration of the preceding vehicle. This radar receives measurements at 10Hz (i.e. 10 per second) and it takes an additional 5ms to process each measurement. The radar measurements consist of the distance to the preceding vehicle and the relative velocity to the preceding vehicle. The acceleration of the preceding vehicle can be computed according to two consecutive measurements of the relative velocity.

In Algorithm 4.2, the pseudocode for the simulation of the ACC controller is given. The delays in processing radar data and braking are hard-coded: for example, if a braking activity occurs, the change in acceleration is then explicitly scheduled for \( t + 0.15 \).
The radars operate asynchronously, which can be seen in the algorithm: a radar measurement is done when \((t + \text{radarOffset}) \mod 0.1 = 0\), with radarOffset in the interval \([0, 0.09]\).

```plaintext
/* \(\Delta t = 0.01\text{s}\) */
foreach timestep t do
    if \((t + \text{radarOffset}) \mod 0.1 = 0\) then
        \(\dot{x}_{\text{relative}} \leftarrow \) do radar measurement ;
        \(\dot{x}_{\text{preceding,now}} \leftarrow \dot{x}_{\text{relative}} + \dot{x}_{\text{self}} ;\)
        schedule measurement processing for \(t + 0.05\);
    if Scheduled measurement \(\dot{x}_{\text{preceding,now}}\) then
        \(\ddot{x} \leftarrow \dot{x}_{\text{preceding,now}} - \dot{x}_{\text{preceding,previous}} ;\)
        Braking activity to achieve \(\dot{x}\);
    if Braking activity to achieve \(\ddot{x}\) then
        schedule change in \(\ddot{x}\) for \(t + 0.15\);
    if Scheduled change in \(\ddot{x}\) then
        change \(\ddot{x}\);
        update \(\dot{x}\) according to \(\ddot{x}\);
        update \(x\) according to \(\dot{x}\);

Algorithm 4.2: The simulation of the ACC controller
```

**CACC1 controller**  Both CACC controllers use direct car-to-car communication to exchange messages containing the car’s acceleration. These messages are sent by each car at a frequency of 10Hz. Sending a message has a delay of 1ms.

The CACC1 controller sends the values from its accelerometer to the following vehicle. Since the delay from the radar processing is no longer present in this controller, this controller should be more responsive than the ACC controller.

In Algorithm 4.3, the pseudocode for the simulation of the CACC1 controller is given. The messages are sent asynchronously, which is hard-coded using a messageOffset in the interval \([0, 0.09]\).

**CACC2 controller**  The CACC2 controller is an extension to the CACC1 controller. This extension makes the CACC2 controller more responsive than CACC1. When a braking activity occurs in a car, it takes 150ms until the car actually decelerates. However, the CACC2 cars have a braking model inside, which estimates the actual deceleration of the car immediately after the braking activity. This means that a braking car can send a message containing an estimation of the deceleration
4.1. Evaluating Adaptive Cruise Control Strategies in Worst-Case Scenarios

/* \( \Delta t = 0.01 \text{s} \) */

foreach timestep \( t \) do
  if \((t + \text{messageOffset}) \times 0.1 == 0\) then
    messageBody = \( \bar{x}_{\text{current}} \);
    follower receives messageBody at \( t + 0.01 \);
    if Received message containing \( \dot{x} \) then
      Braking activity to achieve \( \dot{x} \);
    if Braking activity to achieve \( \dot{x} \) then
      Schedule change in \( \bar{x} \) for \( t + 0.15 \);
    if Scheduled change in \( \dot{x} \) then
      change \( \bar{x} \);
      update \( \dot{x} \) according to \( \bar{x} \);
      update \( x \) according to \( \dot{x} \);
  
Algo\#m 4.3: The simulation of the CACC1 controller

*before* the deceleration actually happens. This results in much faster response times by the following vehicles.

In Algorithm 4.4, the pseudocode for the simulation of the CACC2 controller is given. The messages are sent asynchronously, which is hard-coded using a messageOffset in the interval \([0, 0.09]\).

/* \( \Delta t = 0.01 \text{s} \) */

foreach timestep \( t \) do
  if \((t + \text{messageOffset}) \times 0.1 == 0\) then
    if Scheduled message containing \( \dot{x} \) then
      messageBody = \( \dot{x} \);
    else
      messageBody = \( \bar{x}_{\text{current}} \);
    follower receives messageBody at \( t + 0.01 \);
    if Received message containing \( \dot{x} \) then
      Braking activity with modelled \( \dot{x} \);
    if Braking activity with modelled \( \dot{x} \) then
      Schedule change in \( \bar{x} \) for \( t + 0.15 \);
      Schedule message with \( \dot{x} \);
    if Scheduled change in \( \dot{x} \) then
      change \( \dot{x} \);
      update \( \dot{x} \) according to \( \bar{x} \);
      update \( x \) according to \( \dot{x} \);
  
Algo\#m 4.4: The simulation of the CACC2 controller

51
Chapter 4. Safety in the Face of Uncertainty

The algorithm shows that the braking activity results in a change in acceleration after the mechanical delay of 150 ms, but the message containing the modelled deceleration is scheduled right away and sent to the following vehicle in the next message.

Note that in both CACC algorithms, we use time-based communication as opposed to event-based communication. We chose for time-based communication because it is more robust; a message is sent every 0.1 s, instead of only when something of interest happens (e.g. an emergency brake). This means that cars can expect messages at a regular interval, so they can also notice if a message does not arrive.

Uncertainty in Information and Communication The above algorithms are all not very complicated and one could probably compute the minimum safety distance cars should keep from each other when driving. However, in reality, there is a lot of uncertainty in measurements and communication. The information that the cars use for their calculations are all based on uncertain values and measurements. For example, radars do not perfectly measure the relative speed to the preceding vehicle, it does contain some error. This type of uncertainty is uncertainty in information.

There is also uncertainty in communication: when a message is sent, there is no guarantee that the message will arrive.

We have modelled both these types of uncertainty. Below we describe each piece of uncertain information that we modelled. When we mention a value of $\sigma$, the uncertainty is normally distributed around the correct value $\mu$: $N(\mu, \sigma)$.

- Radar range rate: $\sigma = 0.1 \text{ m/s}$. This is the relative velocity measurement of the radar. This influences the computation of the preceding car’s deceleration in the ACC vehicles;

- Failure in radar range rate: 1 in 1000. The radar fails to measure the relative velocity of the preceding vehicle in 0.1% of the cases;

- Own velocity: $\sigma = 0.1 \text{ m/s}$. This also influences the computation of the preceding car’s deceleration in the ACC vehicles;

- Own max braking power: $\sigma = 0.3 \text{ m/s}^2$ one-sided. This is the error in a car’s estimation of its own maximum braking power. For example, it could be that
4.1. Evaluating Adaptive Cruise Control Strategies in Worst-Case Scenarios

A car thinks it can brake with $-9 \text{m/s}^2$, while in reality this is $8.7 \text{m/s}^2$. It is one-sided, since $-9 \text{m/s}^2$ is a car’s maximum braking power. This influences the CACC2 messages with the modelled braking power;

- Own modeled acceleration: $\sigma = 0.3 \text{m/s}^2$. This is the uncertainty of the estimation of the acceleration when a braking action occurs. This is the value that is sent by the CACC2 vehicles, before the deceleration actually occurs;

- Own accelerometer value: $\sigma = 0.2 \text{m/s}^2$. In the CACC1 controller, the car only sends out the estimation of its own acceleration. This has slightly less uncertainty than the modelled acceleration;

- Failure in broadcasting: 1 in 100. About 1% of sent messages do not arrive at its destination.

- Radar range measurement: $\sigma = 0.5 \text{m}$. The distance to the preceding vehicle. This measurement is currently not used in our simulations, but becomes important when designing the controllers that have to keep a certain safe distance.

These values can be seen as realistic. However, different cars and technologies have different values of uncertainty. Of course, we can change these values for different cars. This would result in different values for minimal safe headway time, as we would run the experiments again, with different parameter values. This problem is tackled in section 4.3, where we provide an analytical method that is able to deal with changes in parameter values in real time.

4.1.4 Experiments and Results

In this subsection, we will describe the experiments that we did and the results that we obtained.

4.1.4.1 Scenario

The objective of our experiments is to determine what the minimal safe headway time is between cars. Our experimental variable therefore is the headway time.

In our scenario, three cars drive on a highway, with varied initial velocities. The first car starts braking as hard as he can on $t = 0$. Then, we observe how the other
cars react and if any crashes occur. In Figures 4.1(a) and 4.1(b), we illustrate this scenario.

Using three cars in this scenario is sufficient for finding the minimal headway time. Using more cars in this scenario has no added value for the results. This is especially the case for the CACC controllers, when cars should be able to communicate their deceleration to more than one follower. While this is not a feature of CACC in this work, we do envision this functionality when further designing these controllers.

The first experimental variable is the type of controller: we did tests with an ACC controller and two different CACC controllers (CACC and CACC2); the second experimental variable is the headway time. We tested each value from 0.05s to 0.7s, with an interval of 0.01s. The third variable is the initial velocity of the cars, that we varied from 20m/s to 40m/s, with an interval of 5m/s. We ran each setting 50 times, resulting in a total of 3 controllers × 66 variations in headway time × 5 different initial velocities × 50 runs per setting = 49,500 runs in total. The main observable is the cumulative number of crashes that occur in each setting.

4.1.4.2 Results

The results are summarised in Figures 4.2, 4.3 and 4.4. Figure 4.2 shows the results for ACC, Figure 4.3 shows the results for the CACC controller and Figure 4.4
4.1. Evaluating Adaptive Cruise Control Strategies in Worst-Case Scenarios

Figure 4.2 – Number of crashes vs. headway time for the ACC controller.

Figure 4.3 – Number of crashes vs. headway time for the CACC1 controller.

Figure 4.4 – Number of crashes vs. headway time for the CACC2 controller.

shows the results for the CACC2 controller. From these graphs, it is apparent that the ACC controller performs the worst, the CACC1 controller is a bit better and the CACC2 controller performs best.
4.1.5 Analysis

From Figures 4.2, 4.3 and 4.4, we can see that the CACC controllers outperform the ACC controller. This is what we expected. The difference can be attributed to the fact that the CACC controller can communicate its deceleration faster and more accurately. The ACC controller can only react on the actual deceleration of its predecessor, which makes it much slower. The CACC2 controller is faster than the CACC1 controller because it communicates its estimated deceleration, before the vehicle actually slows down.

It is nice to see that the number of crashes drops very steeply from a certain value. The crashes with the ACC controller drops between headway times between 0.45 and 0.55 and the crashes with the CACC controller drops between headway times between 0.055 and 0.155.

The most valuable values of these charts are, for each velocity, the lowest headway time at which no crashes occur. These values are plotted in Figure 4.5. This graph can now be used inside the controllers, to determine the preferred headway time to a preceding vehicle. Because the uncertainty in our models did introduce some outliers (see, for example, a crash that occurred with the CACC1 controller at t = 0.61 in Figure 4.3), we left out the most distant 2% of these outliers in this figure. The implication of leaving out these outliers is that, when using Figure 4.5 as a guideline for minimal safety distance, in approximately 2% of all emergency braking situations, a (very soft) crash occurs. We argue that this figure is acceptable, especially when keeping in mind that this figure could have been lower if we would have done more runs.

It may be interesting to compare these numbers with the official guideline. In the Netherlands, the guideline is to keep a headway time of two seconds to the preceding vehicle. However, a comparison with this guideline is a bit unfair, because nobody actually obliges this guideline. As stated in the introduction, current commercially available ACC systems have a minimal headway time of 1 second. When we compare this number to the results from figure 4.5, we see that using a CACC controller, the safe headway time is drastically improved.

The headway times that we found are based on our simulation results, which performed under very specific values of parameters in our model and the uncertainties. This means that when the car model or the uncertainty changes, we would have to rerun the experiments.
We will try to overcome this problem in future work. Ideally, our system would run on-line in a vehicle and is able to adapt to changing situations, such as different weather conditions and malfunctioning sensors. And instead of determining the number of crashes given some headway time, we will approximate the minimal safe headway time given the uncertain parameters and some desired maximum probability that a crash occurs. This will give our work a better theoretical and statistical foundation.

4.1.6 Conclusion

This section describes work into the safety aspect of adaptive cruise control (ACC) and cooperative adaptive cruise control (CACC) strategies, taking into account the uncertainty under which these strategies must operate. We have modelled different strategies, including uncertainty of information and communication in the model. Because we included many different kinds of uncertainty that cars are encountering in real life as well, we can use the found values for safe headway time in the actual implementations of the controllers.

Our experimental results clearly show that the CACC controllers perform better than the ACC controller. We have also shown that the CACC2 controller, that is able to communicate an estimate of its deceleration before the vehicle actually decelerates, performs much better than the CACC1 controller, that is only able to communicate its measured deceleration.
Future Work

Currently, we have only tested our controllers in worst-case scenarios. This means that our controller can be considered only as a safety controller, that knows what the minimal headway time should be given a certain velocity and given uncertainty in parameters.

In the work of Naus et al. (2010), a string stability controller is developed, in which the safety aspect is currently ignored. Therefore, we plan to combine our safety controller with their string stability controller.

Another thing we are planning to do is to do more experiments with the uncertainty in the controllers. Currently, values of the uncertainty are explicitly known to the controllers. However, in many cases, this uncertainty is unknown beforehand or can change over time. We will look at ways in which a controller is able to learn the correct values on the fly. We need on-line machine learning techniques for this, because these values are different for each vehicle and they may even vary on-line as well. This means that a controller must learn these values for each new car that it encounters.
4.2 Approximating Safe Spacing Policies for Adaptive Cruise Control Strategies

In the development of Cooperative Adaptive Cruise Control (CACC) systems, spacing policies are primarily developed for optimisation of string stability and traffic stability. However, the safety issue is hardly taken into account. Uncertainty in the communication network and sensor information makes deciding upon a safe minimal headway a non-trivial task. In this section, we propose a model that is able to approximate the minimal safe headway time, given uncertainty of parameters with varying velocities. By simulating emergency stops, we use the difference in displacement of the cars and a desired maximum probability of a crash to approximate the minimum headway time that yields this probability of a crash. The resulting method is necessary for platooning, a major research development in vehicular networking systems.

4.2.1 Introduction

Cooperative Adaptive Cruise Control (CACC) is one of the major developments in recent research on Vehicular Networking (VN). Its predecessor, i.e. ACC, is currently successfully being deployed commercially by automotive manufacturers and CACC is expected to shortly ship to the market as well. The proposed cooperation in CACC means that vehicles have a wireless vehicle-to-vehicle communication system (which then needs a new control logic) in addition to an existing ACC system.

The main challenge for CACC is to drive comfortably and safely at headway times that are significantly smaller than what is currently possible with human drivers (which is, typically, one second). The headway time is the distance measured in time between vehicles in a transit system. It is one of the pivotal parts

Section 4.2 was published as:

of the so-called *spacing policy* in a cruise control system, which refers to the desired steady state distance that an (C)ACC system attempts to maintain from the preceding vehicle.

With the addition of a communication system, there are also many *uncertainties* introduced in the control system, in addition to the already existing uncertainties from the radar. Such uncertainties can come from, among others, packet loss distributions, vehicle modelling errors and driver behaviour models. These uncertainties need to be incorporated when determining appropriate spacing policies. But there are also specific *dynamics* that we need to consider because of the variety in surrounding vehicles and changing environmental conditions (e.g. road network, weather). The combination of these uncertainties and dynamics make it impossible to completely determine good spacing policies (and headway times) beforehand. We thus need to move to variable (or: dynamic) policies instead of constant ones. We have seen a similar shift of research attention for ACC systems (Swaroop and Rajagopal, 2001) 10-15 years ago, which lead to development of variable-time gap (headway time) policies instead of constant-time gap ones.

Much work in CACC focuses on the comfort issue (e.g. to create string-stable platoons), but we concentrate in this section on the *safety* aspect. We aim to answer the question what are safe headway times *given current circumstances*. These circumstances then refer to the before-mentioned uncertainties and dynamics. In CACC, the coordinated safety management tries to guarantee safety in worst-case future developments (e.g. emergency braking). In this section, we investigate these worst-case scenarios by means of computer simulations based on given (C)ACC models. The considered scenario includes two vehicles that are driving behind each other, where the first vehicle makes an emergency stop – we then look at the reaction of the second vehicle. On a side note, these simulations could be seen as a Monte Carlo algorithm that could later be used on-line (i.e. while the vehicle is driving) to dynamically determine spacing policies, but this is currently outside the scope of this section.

While we thus consider determining headway times on-line as future work, the above arguments should be sufficiently compelling to completely abandon attempting to determine constant times beforehand (i.e. off-line) for CACC systems and we should move to try designing policies with *variable* headway times. In the experiments in this section, we systematically investigate the effects of such variable times in different systems (ACC and two different CACC systems). This is
the necessary ground work that needs to be done before we can move to developing \textit{on-line} methods determining variable-time spacing policies.

In our experimental approach, we assume certain parameters of uncertainty. Using these parameters, we experimentally derive the minimal safe headway time. Our results are therefore only valid for this particular set of uncertainty parameters. However, the main point we make in this section, is that our approach is valid. In other words, our method will derive the minimal safe headway time for a particular set of uncertain parameters and we illustrate this in this section by experimenting with a particular instance of these parameters.

\subsection{4.2.2 Background}

Our work is positioned with (C)ACC systems that address spacing policies based on variable and constant headway times, where performance is measured in terms of safety, comfort and traffic flow improvement. We focus in particular on modelling parametric uncertainties for CACC systems. We briefly overview relevant literature on these topics here.

Petrov (2009) builds a non-linear adaptive tracking controller for a two-vehicle convoy, where the vehicles communicate neither with each other nor with the road infrastructure. Instead, standard robotic methodology is applied to do autonomous vehicle following, combined with a feedback-based controller (employed by the follower vehicle). This work assumes (actually, aims at) a prescribed intervehicle distance (what we call headway time).

An extensive review of constant headway times for ACC is done by Swaroop and Rajagopal (2001). Three different performance criteria are considered: stability, safety and traffic flow behaviour. For ACC, safety guarantees can be given, even such that errors in spacing do not amplify. Concerning stability and flow, smaller headway times are required to achieve higher throughputs. The review also shows that the control effort of an ACC system with a constant headway time is inversely proportional to headway time: the smaller this time, the greater the control effort.

Parametric uncertainties in ACC systems have been researched by Swaroop and Hedrick (1994). These uncertainties concern vehicle mass, aerodynamic drag and time drag. The provided solution to address these uncertainties is a Lyapunov-based decentralised adaptive control algorithm.
Santhanakrishnan and Rajamani (2003) have developed a framework for design and evaluation of spacing policies for ACC. Although the evaluation criteria include string and traffic flow stability and traffic flow capacity, the framework does not explicitly address safety.

Safety in ACC is an issue that is addressed explicitly by Wang and Rajamani (2004a). In this work, an ACC system is proposed that can improve traffic flow and ensure safe operation. The novelty of the system is that it uses a new inter-vehicle spacing policy, in which the spacing is a non-linear function of vehicle speed (called the variable time-gap, VTG, policy). In comparison with a (then) traditional constant time-gap, CTG, policy, the same level of safety is provided, while improving the traffic flow. The question if ACC systems should in general be designed to maintain a constant time-gap between vehicles, is addressed in Wang and Rajamani (2004b). Another approach that improves CTG based systems is described in Zhao et al. (2009). This article demonstrates a new spacing policy that is safe and improves traffic flow. The policy is a non-linear function of vehicle velocity and uses the vehicle state and braking capacity information. The policy works best in high-density traffic conditions.

Yi and Horowitz (2006) propose an approach to macroscopic traffic flow propagation stability for ACC vehicles. In this approach, a non-linear traffic flow stability criterion is used with a wave-front expansion technique. In earlier approaches, a macro- with microscopic model was necessary with a constant headway time. The new approach covered all stability conditions obtained for these earlier approaches. Another VTG-policy based ACC system is proposed by Zhang and Ioannou (2005). This control system guarantees stability and it regulates speed and separation errors toward zero (with the leading vehicle drives a constant speed).

While ACC systems are currently being adopted in consumer vehicles, research and development into cruise control focuses on enabling more and better cooperation between ACC systems, yielding so-called CACC systems. Van Arem et al. (2006) describe the effect of CACC on traffic flow. They conclude that, when the penetration level of CACC-equipped vehicles is high enough (> 60%), traffic stability and throughput is improved. In Yang et al. (2004), a communication protocol is proposed in order to make a cooperative collision warning system on highways.

The main application area of CACC technology these days is platooning. Broggi et al. (2000) and Kanellakopoulos et al. (1999) both use image recognition techniques in combination with sensors to autonomously enable platooning. However,
current technology has improved significantly since then and nowadays direct radio communication between vehicles is used to enable platooning.

Naus et al. (2010) thoroughly investigate the issue of string stability in platooning, with both ACC and CACC controllers. Their method includes several factors of delay in communication, but uncertainty of the information is not taken into account.

In Hallé (2005), an extensive architecture is given for a layered multi-agent CACC architecture. The authors use this architecture to implement both centralised platoons (in which there is a coordinating platoon leader) and decentralised platoons (in which all cars operate as equals). Khan et al. (2008) present different platoon (in their paper, convoy) forming strategies, based on a utility value of a platoon.

To summarise, in all of the above approaches to designing CACC systems, uncertainty in information and communication is not accounted for. Also, the works focus on comfort (string stability) rather than safety. While these points have been addressed for ACC (as shown above in the first part of this subsection), this has not been picked up in CACC development and research. These are the points that we address in this section: safety and parametric uncertainty in CACC systems. We build further on earlier work (van Willigen et al., 2011a) where minimal safe headway times were experimentally determined for a number of different (C)ACC controllers.

4.2.3 Model

In this subsection, we describe the model that we used for our simulations. This model is an extended version of the model that we introduced in van Willigen et al. (2011a). It is a numerical model in which different kinds of uncertainty are explicitly modelled. This presence of uncertainty in information and communication justifies our choice for simulation, since it makes a mathematical analysis of the problem too complex.

First we describe how we modelled the cars in our simulation and second, we describe the uncertainty in information and communication that we included in our model.


4.2.3.1 Cars

We do experiments with three different types of car controller. First, there is the adaptive cruise control (ACC) controller. This controller uses the radar sensor of the car to derive information about distance, velocity and acceleration from the preceding vehicle. Second and third are the two cooperative adaptive cruise control controllers (CACC\textsubscript{1} and CACC\textsubscript{2}). These controllers make use of direct communication between vehicles to derive the same information about preceding vehicles. CACC\textsubscript{1} communicates current state information, whereas CACC\textsubscript{2} communicates intended state information (more about this later).

The difference in technology between the ACC controller and the CACC controllers has some implications. First, a radar sensor can only measure the range (direct distance to the preceding vehicle) and range rate (relative velocity of the preceding vehicle). This means that information about the acceleration of the preceding vehicle needs to be derived from this information. When using direct communication, information about acceleration can be transmitted directly. Second, information that a car obtains from its own sensors (e.g. wheel encoders for velocity, accelerometer for acceleration) is more accurate than information a car derives from its radar input.

The task of our safety controller is to determine what the minimal safe headway time is, with the constraint that it still must be safe. We define the minimal safe headway time as follows: it is the headway time a car must keep from its preceding vehicle at which the probability of a crash does not exceed 0.1\%. This definition can easily be changed to a lower or higher number, using the same model. We get back to this in Section 4.2.4.2.

All controllers share the same update scheme, that is based on a simple physics model in which velocity is updated according to acceleration, after which position is updated according the velocity. This update scheme is depicted in Algorithm 4.5. Note that in our models we denote velocity as the first derivative of position, $\dot{x}$ and the acceleration as the second derivative of position, $\ddot{x}$.

```
/* $\Delta t = 0.01$ */
foreach timestep $t$ do
    $\ddot{x}_t \leftarrow$ compute new $\ddot{x}$;
    $\dot{x}_t \leftarrow \dot{x}_{t-1} + \ddot{x}_t \Delta t$;
    $x_t \leftarrow x_{t-1} + \dot{x}_t$;

Algorithm 4.5: Update scheme for cars
```
All controllers use this simple update scheme. The difference between the ACC and the two CACC controllers lies in the way the controller derives the acceleration of the preceding vehicle. The ACC controller does this using radar, while the CACC controllers directly transmit information about their own acceleration to the following vehicle. In the following paragraphs, the controllers are described in detail. These descriptions are at the agent level.

**ACC controller** The radar in the ACC controller has an update rate of 10Hz and it takes an additional 5ms to process each measurement. Each measurement consists of two values: the range (i.e. the distance to the preceding vehicle, measured in meters) and range rate (i.e. the relative velocity to the preceding vehicle). Using two consecutive range rate measurements, the acceleration of the preceding vehicle can be derived.

In Algorithm 4.6, the pseudocode for the behaviour of this controller is given.

```
foreach timestep t do
    if Processed radar measurement m then
        \( \dot{x}_{\text{preceding},t} \leftarrow m.\text{rangeRate} + \dot{x}_{\text{self}}; \)
        \( \ddot{x}_{\text{preceding},t} \leftarrow \dot{x}_{\text{preceding},t} - \dot{x}_{\text{preceding},t-1}; \)
    changeAcceleration(\( \ddot{x}_{\text{preceding},t}; \))

Algorithm 4.6: Behaviour of the ACC controller
```

This algorithm takes the range rate from each radar measurement, computes the acceleration of the preceding vehicle and changes its own acceleration according to this acceleration. No communication between vehicles occurs in this setting.

**CACC\(_1\) controller** Both CACC controllers use direct vehicle-to-vehicle (V2V) communication to exchange messages contains the vehicle’s acceleration. These messages are sent asynchronously by each car at a frequency of 10Hz.

The CACC\(_1\) controller sends the measured value from the accelerometer to the following vehicle. This means that the value in this message can directly be used by the following vehicle, instead of having to derive the acceleration from multiple radar readings. This makes the CACC\(_1\) controller more responsive than the ACC controller. In Algorithm 4.7, the pseudocode for the behaviour of this controller is given.

This algorithm copies the acceleration value that it received from the preceding vehicle. Also, this controller makes sure that that each car sends messages con-
Chapter 4. Safety in the Face of Uncertainty

foreach timestep \( t \) do

if Received message \( m \) then // from preceding car

\[ \dot{x}_{\text{preceding},t} \leftarrow m.\dot{x}; \]

changeAcceleration(\( \dot{x}_{\text{preceding},t} \));

if Sender is ready then // 10Hz // to following car

sendMessage(\( \dot{x}_{\text{actual}} \));

Algorithm 4.7: Behaviour of the CACC\( _1 \) controller

taining its own acceleration (measured from the accelerometer) to the following vehicle.

CACC\( _2 \) controller The CACC\( _2 \) controller is a simple extension of the CACC\( _1 \) controller. Instead of the value of the accelerometer, the predicted value of the acceleration is communicated to the following vehicle. Since there is a delay between a braking action and the actual deceleration of the vehicle of 150ms and the vehicle is able to predict the actual deceleration very well at the time of the braking action, this is a very sensible thing to do.

In Algorithm 4.8, the pseudocode for the behaviour of this controller is given.

foreach timestep \( t \) do

if Received message \( m \) then // from preceding car

\[ \dot{x}_{\text{preceding},t} \leftarrow m.\dot{x}; \]

changeAcceleration(\( \dot{x}_{\text{preceding},t} \));

if Sender is ready then // 10Hz // to following car

sendMessage(\( \dot{x}_{\text{predicted}} \));

Algorithm 4.8: Behaviour of the CACC\( _2 \) controller

This algorithm is very similar to the CACC\( _1 \) algorithm. The only subtle difference is that, instead of the measured acceleration from the accelerometer, the predicted acceleration is communicated the following vehicle. This makes the CACC\( _2 \) controller more responsive than the CACC\( _1 \) controller.

Both CACC controllers use time-based communication as opposed to event-based communication. On the one hand, this makes our model more robust, since receiving vehicles know when to expect new messages and are able to anticipate when a message does not arrive. This is not the case when we use event-based communication. On the other hand, time-based communication is less scalable.
When many cars within a certain range are constantly broadcasting messages, this will influence the quality of the network.

4.2.3.2 Uncertainty and Delay

The algorithms that we described in the previous subsections are fairly straightforward and if one knows the delays of communication and information processing, it would seem easy to compute the minimal safe headway time cars should maintain to avoid crashes. However, since sensor information and wireless communication comes with a lot of uncertainty, the computation of the minimal safe headway time is not a trivial task. In this subsection, we describe the various factors of uncertainty that are present in our sensors and the values we used for these uncertain parameters. These values can be seen as realistic. However, if the values of these parameters change, we can still use the same model to approximate the safe headway time that belongs to that parameter set. We will discuss this issue further in Section 4.2.5.

We have modelled all uncertainties using a Gaussian distribution with deviation $\sigma$ from the correct value $\mu$. The delays are hard-coded in the model.

Uncertainty:

- Radar range rate: $\sigma = 0.1\text{m/s}$. This is the relative velocity measurement of the radar. This influences the computation of the preceding car’s deceleration in the ACC vehicles;

- Failure in radar range rate: in 0.1% of the radar measurements, the radar fails to measure the relative velocity of the preceding vehicle;

- Own velocity: $\sigma = 0.1\text{m/s}$. This also influences the computation of the preceding car’s deceleration in the ACC vehicles;

- Own max braking power: $\sigma = 0.3\text{m/s}^2$ one-sided. This is the error in a car’s estimation of its own maximum braking power. For example, it could be that a car thinks it can brake with $-9\text{m/s}^2$, while in reality this is only $8.7\text{m/s}^2$. It is one-sided, since $-9\text{m/s}^2$ is a car’s maximum braking power. This influences the CACC2 messages with the predicted braking power;

- Own predicted acceleration: $\sigma = 0.3\text{m/s}^2$. This is the uncertainty of the estimation of the acceleration when a braking action occurs. This is the value that is sent by the CACC2 vehicles, before the deceleration actually occurs;
Chapter 4. Safety in the Face of Uncertainty

- Own accelerometer value: $\sigma = 0.2\text{m/s}^2$. In the CACC\textsubscript{1} controller, the car only sends out the estimation of its own acceleration. This has slightly less uncertainty than the predicted acceleration;

- Failure in broadcasting: About 1\% of all sent messages do not arrive at their destination. This is a simplification, because currently, we do not take into account bursts of packet loss.

- Radar range measurement: $\sigma = 0.5\text{m}$. The distance to the preceding vehicle. This measurement is not used in our simulation, but we do include it in our calculations to determine the minimal safe distance. See Section 4.2.5 for more detail.

Delay:

- Mechanical brake delay: 150ms. This is the delay between the change in acceleration and the actual start of that acceleration. This is essentially the difference between CACC\textsubscript{1} and CACC\textsubscript{2}.

- Radar processing delay: 50ms. This is how long it takes before a radar signal is processed and ready to use.

- Communication delay: 10ms. This is how long it takes for a message to be received.

These are the uncertainties and delays that we incorporated in our experiments. Others include delay in bus, gateway and radio channel access. These are not included in our experiments, but these could easily be incorporated.

4.2.4 Experiments and Results

In this subsection, we describe the scenario that we implemented and the experiments we performed in this scenario.

4.2.4.1 Scenario

The objective of our experiments is to find a minimal safe spacing policy for different adaptive cruise control controllers, given various parameters that describe factors of uncertainty in the system, as well as the vehicle’s velocity.
To this end, we simulate 2 cars on a highway, of which the front vehicle applies an emergency brake at $t_0$. The following car has to respond to this emergency brake. The way the following car responds to the emergency brake, depends on the controller in the car. The ACC controller uses radar, the CACC1 controller uses direct communication of the actual deceleration and the CACC2 controller uses direct communication of the predicted deceleration. In Figures 4.11(a) and 4.11(b), these scenarios are illustrated.

In our experiments, we use a homogeneous set of cars. This is a simplification of real situations on highways, where there are of course many different kinds of vehicles with different capabilities. This model can however be used with any vehicle model and any model of uncertainty. Our experiments are an example of how this model can be used to determine minimal safe headway times.

We did simulations with two different initial velocities, 20m/s and 30m/s. We then measured the difference in displacement:

$$\Delta s_{j,i} = s_j - s_i$$

(4.1)

where $s_i$ is the displacement of the front vehicle and $s_j$ the displacement of the following vehicle.

The displacement of a vehicle $i$ is the distance that $i$ travelled from $t_0$ to the moment of standstill $t_{ss}$:

$$s_i = x_{i,t_{ss}} - x_{i,t_0}$$

(4.2)

We did experiments with 3 controllers (ACC, CACC1 and CACC2) $\times$ 2 initial velocities (20m/s and 30m/s) $\times$ 100,000 runs per setting = 600,000 runs in total.

For all these runs, we measure the difference in displacement of the cars, $\Delta s$. For each combination of a controller with an initial velocity, we then obtain the distributions of $\Delta s$. For the remainder of this section, we will call these the base distributions $B_{c,\dot{x}}$, with $c \in \{\text{ACC, CACC1, CACC2}\}$ and $\dot{x}$ being the initial velocity of the vehicles and in our experiments, $\dot{x} \in \{20, 30\}$.

### 4.2.4.2 Safe spacing policy approximation

In this subsection, we explain how we can use the base distributions $B_{c,\dot{x}}$ to approximate the minimal safe headway time. In order to make these calculations, some variables must be introduced:
Chapter 4. Safety in the Face of Uncertainty

(a) The ACC scenario, in which the following car uses its radar to obtain information about relative position and velocity of the car in front.

(b) The CACC scenario, in which the car in front uses direct communication to transmit its acceleration to the following vehicle.

Figure 4.6 – The difference between the ACC and CACC scenario.

- $h_{\dot{x}, \text{safe}}$: the minimal safe headway time, given velocity $\dot{x}$. This is what we want to compute;
- $p(\text{Crash})$: the acceptable probability for a crash. We use a value of 0.1%;
- $\Delta s_{\dot{x}, \text{safe}}$: the safe difference in displacement, given velocity $\dot{x}$. This value depends on $\dot{x}$ and on the value of $p(\text{Crash})$;
- $d_{i,j}$: the final distance between vehicle $i$ and $j$ at standstill, after a run;

For each distribution $B_{\dot{x}, \dot{\dot{x}}}$, we can compute the value of $\Delta s_{\text{safe}}$. In order to do this, we first compute the convolution of the base distributions with the uncertainty on the radar range. This is necessary, because the safe distance is dependent on how well a car is able to determine its distance to the preceding vehicle. Then, we determine $\Delta s_{\text{safe}}$ by looking up at which value of $\Delta s$, the right-hand side tail of the distribution consists of $p(\text{Crash})\%$ of the runs. In our experiments, we set the desired probability of a crash at 0.1%, but this can easily be set to a different value without having to redo the experiments. In this case, only this final step needs to be recalculation.

Using these values, we can now compute the minimal safe headway time $h_{\dot{x}, \text{safe}}$, by dividing these values by the initial velocity of the vehicles.

$$h_{\dot{x}, \text{safe}} = \frac{\Delta s_{\dot{x}, \text{safe}}}{\dot{x}}$$ (4.3)
4.2. Approximating Safe Spacing Policies for Adaptive Cruise Control Strategies

Table 4.1 – The values of $\Delta s_{\text{safe}}$ for each base distribution $B$.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$\Delta s_{\text{safe}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_{\text{ACC,20}}$</td>
<td>10.53m</td>
</tr>
<tr>
<td>$B_{\text{CACC1,20}}$</td>
<td>7.91m</td>
</tr>
<tr>
<td>$B_{\text{CACC2,20}}$</td>
<td>4.81m</td>
</tr>
<tr>
<td>$B_{\text{ACC,30}}$</td>
<td>17.42m</td>
</tr>
<tr>
<td>$B_{\text{CACC1,30}}$</td>
<td>12.17m</td>
</tr>
<tr>
<td>$B_{\text{CACC2,30}}$</td>
<td>7.69m</td>
</tr>
</tbody>
</table>

Table 4.2 – The values of $h_{\dot{x},\text{safe}}$ for each base distribution $B$.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$h_{\dot{x},\text{safe}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_{\text{ACC,20}}$</td>
<td>0.53s</td>
</tr>
<tr>
<td>$B_{\text{CACC1,20}}$</td>
<td>0.40s</td>
</tr>
<tr>
<td>$B_{\text{CACC2,20}}$</td>
<td>0.24s</td>
</tr>
<tr>
<td>$B_{\text{ACC,30}}$</td>
<td>0.58s</td>
</tr>
<tr>
<td>$B_{\text{CACC1,30}}$</td>
<td>0.41s</td>
</tr>
<tr>
<td>$B_{\text{CACC2,30}}$</td>
<td>0.26s</td>
</tr>
</tbody>
</table>

4.2.4.3 Results

The results of our simulations are visualised in Figures 4.7 and 4.8, that show respectively the experiments with initial velocity $\dot{x} = 20\text{m/s}$ and $\dot{x} = 30\text{m/s}$. The further to the right on the horizontal axis the distributions are, the higher the distance is that the following vehicle has travelled, relative to the leading vehicle. So, the further the distributions are to the left on the horizontal axis, the lower the minimal safe headway time will be for the following vehicles.

Using the safe spacing policy approximation method as described in the previous subsection, we obtained for each base distribution $B$ the safe distance $\Delta s_{\dot{x},\text{safe}}$ and the safe headway time $h_{\dot{x},\text{safe}}$ given a certain initial velocity $\dot{x}$. These results are summarised in Tables 4.1 and 4.2.

4.2.5 Analysis

In this subsection, we analyse the results that we presented in the previous subsection and we validate these results by doing control experiments with the results that we found.
Chapter 4. Safety in the Face of Uncertainty

Figure 4.7 – Results for our controllers when using initial velocity $v = 20\text{m/s}$.

Figure 4.8 – Results for our controllers when using initial velocity $v = 30\text{m/s}$.
4.2. Approximating Safe Spacing Policies for Adaptive Cruise Control Strategies

4.2.5.1 Table analysis

The main results of our work are summarised in Tables 4.1 and 4.2. We see that the differences in safe headway time are very small for different initial velocities. This means that safe headway time is (more or less) independent of initial velocity.

The results for safe headway time are very promising in terms of feasibility. The numbers that we obtained are the absolute minimum headway time that cars should keep. However, research into string stability has shown that ideal headway times are usually larger than the numbers that we found. For example, research on string stability by Naus et al. (2010) report on minimal headway times of 2.8s for ACC systems and 0.8s for CACC systems. These numbers are much bigger than our results for the safety controller. This is good news, because in this case, the string stability controller will not often be interfered by our safety controller.

4.2.5.2 Base distribution analysis

In Figures 4.7 and 4.8, we show the histograms that contain the distributions of our measured variable, $\Delta s$. These are the base distributions $\mathcal{B}_{c,x}$, that we introduced in Section 4.2.4. The higher the $\Delta s$, the higher the relative displacement of the following vehicle. This means that lower values for $\Delta s$ are better.

On an abstract level, we can draw the following conclusions about the controllers:

- As expected, the ACC controller performs the worst and the CACC2 controller performs the best.

- There is a difference in the shape of the distributions between the ACC controller and the two CACC controllers: The CACC distributions are flatter than the ACC distribution. This is due to the uncertainty and delay that are introduced with wireless communication.

- When comparing Figure 4.7 to Figure 4.8, we see a relation between the distributions of the same controller. The means and standard deviations are both larger for a larger initial velocity. However, we tried to find a direct relation between these distributions, but we could not find such a relation. Further research is needed in order to achieve this. This means that as of now, we cannot create a base distribution in which we leave out the initial velocity as well.
Table 4.3 – Crash statistics in our control experiments.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>% crash</th>
<th>mean crash velocity</th>
<th>max crash velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_{\text{ACC,20}}$</td>
<td>0.09%</td>
<td>1.66m/s</td>
<td>3.02m/s</td>
</tr>
<tr>
<td>$B_{\text{CACC,20}}$</td>
<td>0.07%</td>
<td>2.07m/s</td>
<td>4.60m/s</td>
</tr>
<tr>
<td>$B_{\text{CACC,20}}$</td>
<td>0.1%</td>
<td>1.55m/s</td>
<td>3.55m/s</td>
</tr>
<tr>
<td>$B_{\text{ACC,30}}$</td>
<td>0.09%</td>
<td>2.53m/s</td>
<td>5.50m/s</td>
</tr>
<tr>
<td>$B_{\text{CACC,30}}$</td>
<td>0.07%</td>
<td>1.42m/s</td>
<td>2.95m/s</td>
</tr>
<tr>
<td>$B_{\text{CACC,30}}$</td>
<td>0.1%</td>
<td>1.80m/s</td>
<td>3.39m/s</td>
</tr>
</tbody>
</table>

4.2.5.3 Validation

In this subsection, we will validate the results in Table 4.2. We will do this by running our simulations once again, but now using the initial headway values from that table. In Figure 4.9, the histograms of the final distances between the two vehicles are shown. Negative values on the horizontal axis denote that a crash has occurred. From these histograms, it becomes immediately apparent that only a very small portion of the runs result in a crash.

Table 4.3 shows statistics about the crashes that occurred in these simulations. We measured the percentage of crashes that occurred and we measured the mean velocity of the crashes, as well as the velocity of the fastest crash in each setting. From these results, we see that for all controllers, the probability for a crash is indeed 0.1% or even a little bit less. This validates the values for safe headway time that we obtained from the base distributions $B$.

Apart from the probability for a crash, we also recorded the mean and maximum velocity of the crashes. We can conclude from this table that none of the crashes that occurred in the control experiments are lethal. The worst crash that we found was 5.5m/s. This means that at the time of the crash, the following car was driving 5.5m/s faster than the preceding vehicle. This type of crash is easily damped by seatbelts and the airbag.

4.2.6 Conclusion

In this section, we looked at safe minimal headway times for (Cooperative) Adaptive Cruise Control (CACC) systems. Whereas much research effort is spent on making this system such that they are comfortable for the driver (e.g. that the vehicle does not continuously and abruptly accelerates and brakes), surprisingly less effort goes into making these systems safe.
4.2. Approximating Safe Spacing Policies for Adaptive Cruise Control Strategies

One of the major development transitions in road of commercialising ACC systems, was that it was shown that variable headway times had to be determined dynamically instead of beforehand. In this light, an important contribution of this section is that it demonstrates that the same holds for CACC systems. Moreover, because of the many uncertainties involved with communicating information between vehicles, the urge for dynamic variable headway times is even stronger.

Our investigation has been experimental by means of computer simulation. These experiments show that the resulting safe headway times are more or less independent of initial velocity. Also, the found values for minimal safe headway time are lower than the values that are currently used in controllers that optimise on string stability. This means that our safety controller will not interfere with the string stability controller under normal circumstances.

Our resulting safe headway times still have a small, calculated risk of crashing. But even in the very unlikely event of a crash ($< 0.1\%$), the crashes that occur are far from lethal.

**Figure 4.9** – Results of our validation experiments. Using the safe headway time values from Table 4.2, these are the distributions of final distances from our simulations.
The work described in this section is a first step towards enabling vehicles to fully determine (variable) spacing policies on-line (i.e. while driving) and autonomously. For future work, the next step is to translate the findings of our experimental investigation into letting vehicles determine headway times while driving. On the experimental side, our simulations were designed with minimal requirements for the strict purpose of isolating the investigation of the research question. Still, we also propose scaling up the simulations in the future in terms of numbers of vehicles and complexity of the road network.
4.3 Critical Headway Estimation in Cooperative Adaptive Cruise Control

This section proposes a safety check extension to Adaptive Cruise Control systems where the critical headway time is estimated in real time. This critical headway time estimate enables automated reaction to crisis circumstances such as when a preceding vehicle performs an emergency brake. We introduce a method for critical headway approximation that can handle uncertainty in vehicle state, vehicle behaviour and communication in real time. We validate our method using Monte Carlo simulations where we simulate emergency braking situations to ascertain the safe headway.

4.3.1 Introduction

In Europe’s major cities, the average driver loses as much as 60 to 70 hours per year in traffic jams, the cause of tremendous environmental and socio-economical impact. Vehicle manufacturers are introducing Adaptive Cruise Control (ACC) to help combat this issue. ACC allows the driver to set the desired distance to the preceding vehicle. The ACC system measures the vehicle’s relative distance and velocity through radar and adjusts speed accordingly.

Such automated maintenance of distance is expected to increase the uniformity of traffic flows and so better utilise road capacity. Cooperative Adaptive Cruise Control (CACC) is a major development in recent research on intelligent transportation systems that takes this automated control to the next level by adding direct communication between vehicles. This enables the vehicles to communicate about their current state and behaviour. Thus, it reduces the uncertainty inherent in relying only on radar measurements. Directly communicating such accurate

---


Section 4.3 has been accepted for publication as:

Chapter 4. Safety in the Face of Uncertainty

state information allows vehicles to drive much closer to each other without compromising safety.

Current research in CACC systems focuses mainly on control-theoretic aspects of these systems. One of the big selling points of CACC is that it greatly improves the capacity of highways by allowing rows of vehicles to drive with very low inter-vehicle distances while still being string stable: changes in acceleration or deceleration are reduced by the following vehicles instead of amplified. This property is expected to greatly improve the throughput of vehicles on highways, because it is the amplification of acceleration and deceleration that causes many traffic jams (see Naus et al. (2010)).

Of course, one must consider the safety of (C)ACC systems: the vehicles must maintain a minimum headway time to allow a timely response to any decelerations in the preceding vehicle. Current practice is to maintain a fixed minimum headway time that provides enough of a margin to cater for normal driving conditions. This paper proposes an additional safeguard that initiates forceful deceleration (outside of the \([-3,3]\) range that is typically considered comfortable) when the headway time becomes less than a critical safety threshold, for instance when the preceding vehicle initiates an emergency stop.

Taking uncertainty into account is crucial when considering critical headway times or other safety checks: radars have uncertainty in their readings, vehicles have uncertainty about their state and behaviour, and communication between vehicles can fail. The critical headway time depends on these uncertainties. For example, larger measurement errors and higher probability of communication failure imply a larger critical headway time. Therefore, the critical headway time must be updated continuously and in real time to reflect the current uncertainties.

The purpose of this paper is twofold: firstly, we propose a method that solves the problem of continuously estimating the critical headway time in the face of these uncertainties. Secondly, we want to emphasise the importance for intelligent transportation systems to explicitly take uncertainty into account. We propose a method that uses numerical methods to approximate the critical headway time on the basis of uncertainty in vehicle state, vehicle behaviour and communication.

We apply this method for three different types of adaptive cruise control: when only radar is used to determine the velocity and acceleration of a preceding vehicle (which we call \(e\text{ACC}\)); when the current velocity and acceleration is communicated between vehicles (called \(e\text{CACC}_1\)), and when the current velocity and acceleration
and intended acceleration is communicated between vehicles (eCACC). The e in these cases stands for emergency. We validate the proposed method with a Monte Carlo simulation of a vehicle performing an emergency brake and the subsequent reaction by the following vehicle.

### 4.3.2 Related Work

Our work is positioned with (C)ACC systems that address spacing policies based on variable and constant headway times, where performance is measured in terms of safety, comfort and traffic flow improvement. We focus in particular on modelling parametric uncertainties for CACC systems. We briefly overview relevant literature on these topics here.

Petrov (2009) builds a non-linear adaptive tracking controller for a two-vehicle convoy, where the vehicles communicate neither with each other nor with the road infrastructure. Instead, standard robotic methodology is applied to do autonomous vehicle following, combined with a feedback-based controller (employed by the follower vehicle). This work assumes (actually, aims at) a prescribed inter-vehicle distance (what we call headway time).

An extensive review of constant headway times for ACC was performed by Swaroop and Rajagopal (2001). Three different performance criteria are considered: stability, safety, and traffic flow behaviour. For ACC, safety guarantees can be given, even such that errors in spacing do not amplify. Concerning stability and flow, smaller headway times are required to achieve higher throughputs. The review also shows that the control effort of an ACC system with a constant headway time is inversely proportional to the headway time itself: the smaller this time, the greater the control effort.

Swaroop and Hedrick (1994) also investigated parametric uncertainties in ACC systems. These uncertainties concern vehicle mass, aerodynamic drag and time drag. The provided solution to address these uncertainties is a Lyapunov-based decentralised adaptive control algorithm.

Santhanakrishnan and Rajamani (2003) have developed a framework for design and evaluation of spacing policies for ACC. Although the evaluation criteria include string and traffic flow stability, and traffic flow capacity, the framework does not explicitly address safety.
Safety in ACC is an issue that is addressed explicitly by Wang and Rajamani (2004a). In this work, an ACC system is proposed that can improve traffic flow and ensure safe operation. The novelty of the system is that it uses a new inter-vehicle spacing policy, in which the spacing is a non-linear function of vehicle speed (called the variable time-gap, VTG, policy). In comparison with a (then) traditional constant time-gap, CTG, policy, the same level of safety is provided, while improving the traffic flow. The question if ACC systems should in general be designed to maintain a constant time-gap between vehicles, is addressed by Wang and Rajamani (2004b). This research, although specifically addressing safety, does not take uncertainty into account. Another approach that improves CTG based systems is described by Zhao et al. (2009), who demonstrate a new spacing policy that is safe and improves traffic flow. The policy is a non-linear function of vehicle velocity and uses the vehicle state and braking capacity information. The policy works best in high-density traffic conditions.

Yi and Horowitz (2006) propose an approach to macroscopic traffic flow propagation stability for ACC vehicles. In this approach, a non-linear traffic flow stability criterion is used with a wave-front expansion technique. In earlier approaches, a macro- with microscopic model was necessary with a constant headway time. The new approach covered all stability conditions obtained for these earlier approaches. Another VTG-policy based ACC system is proposed by Zhang and Ioannou (2005). This control system guarantees stability, and it regulates speed and separation errors toward zero (with the leading vehicle driving at a constant speed).

While ACC systems are currently being adopted in consumer vehicles, research and development into cruise control focuses on enabling more and better cooperation between ACC systems, yielding so-called CACC systems. Van Arem et al. (2006) describe the effect of CACC on traffic flow. They conclude that, when the penetration level of CACC-equipped vehicles is high enough (> 60%), traffic stability and throughput is improved. Yang et al. (2004) propose a communication protocol in order to make a cooperative collision warning system on highways.

The main application area of CACC technology these days is platooning. Broggi et al. (2000) and Kanellakopoulos et al. (1999) both use image recognition techniques in combination with sensors to autonomously enable platooning. However, current technology has improved significantly since then, and nowadays direct radio communication between vehicles is used to enable platooning.
4.3. Critical Headway Estimation in Cooperative Adaptive Cruise Control

Naus et al. (2010) thoroughly investigate the issue of string stability in platooning, with both ACC and CACC controllers. Their method includes several factors of delay in communication, but uncertainty of the information is not taken into account.

Hallé (2005) give an extensive architecture for a layered multi-agent CACC architecture. The authors use this architecture to implement both centralised platoons (in which there is a coordinating platoon leader) and decentralised platoons (in which all vehicles operate as equals). Khan et al. (2008) present different platoon (in their paper, convoy) forming strategies, based on a utility value of a platoon.

To summarise, in all of the above approaches to designing CACC systems, uncertainty in vehicle state, vehicle behaviour and communication is not explicitly taken into account. Also, they focus on comfort (string stability) rather than safety. While these points have been better addressed for ACC (as shown above in the first part of this section), this has not been picked up in CACC development and research. Therefore, the main point that we address in this section is the influence of uncertainty in vehicle state, vehicle behaviour and communication on safety in (C)ACC systems. We build further on earlier work (van Willigen et al., 2011a; van Willigen et al., 2011), where we developed a Monte Carlo simulation to solve the problem. This approach was very slow, and therefore it inspired us to develop a solution that could be used by vehicles in real time. The analytical method that we present in this section is now validated by this Monte Carlo approach.

4.3.3 Method

In this section, we describe the method for our safety checker. It is an analytical method that calculates the critical safe headway time while explicitly taking uncertainty into account. This novel method can be very useful in practice because it can update the critical headway time estimate in real time, adapting to changing values of the system parameters and their uncertainties.

Our method can be used in vehicles with different capabilities, and we consider three possible cases. First, there are the capabilities that fit the adaptive cruise control (ACC) controller: this controller uses the radar sensor of the vehicle to derive information about the preceding vehicle’s state and behaviour, such as its velocity and acceleration. Second and third are the capabilities that fit the two cooperative
adaptive cruise control controllers (CACC\textsubscript{1} and CACC\textsubscript{2}). These controllers make use of direct communication between vehicles to obtain this information about preceding vehicles. We name our emergency safety checkers after the capabilities that they correspond with: eACC, eCACC\textsubscript{1} and eCACC\textsubscript{2}.

The difference in technology between the capabilities of eACC and the two versions of eCACC has some implications. First, a radar sensor can only measure the range (direct distance to the preceding vehicle) and range rate (relative velocity of the preceding vehicle). This means that the acceleration of the preceding vehicle needs to be derived from this information. When using direct communication, information about acceleration can be transmitted directly, which is much faster. Second, information from a vehicle’s own sensors (e.g. wheel encoders for velocity, accelerometer for acceleration) is typically more accurate than information a vehicle derives from its radar input. The difference between eCACC\textsubscript{1} and eCACC\textsubscript{2} is that in eCACC\textsubscript{1}, the current state information is transmitted, whereas in eCACC\textsubscript{2}, the intended state information is transmitted.

In this study, we define the critical headway time as follows: it is the headway time a vehicle must keep to ensure that the likelihood of coming into contact with the lead vehicle when reacting automatically to an emergency stop is less than 0.135\% ($3\sigma$ from the mean difference in displacement after both vehicles come to a standstill). This definition of safety can easily be modified to meet other requirements: it is an adjustable parameter of the method we propose.

4.3.3.1 Vehicle Reactions

In this section, we describe how we modelled the vehicles and their reaction to an emergency braking action. We define some analytical expressions that we need in order to deal with uncertain variables.

**Generic Vehicle Braking Model** The model of a vehicle $c$ can be described by a single equation, that describes the reaction of a vehicle to change in acceleration as a function of time $t$ (for more details, see Naus et al. (2010)):

$$a_c(t) = a_c(0) + \left(1 - e^{-t/\tau_c}\right)(a_{c,int} - a_c(0))$$

with $a_c(0)$ the acceleration at $t = 0$, $a_{c,int}$ the intended acceleration and $\tau$ a time constant that tells us how quickly vehicle $c$ reaches the intended acceleration.
Critical Headway Estimation in Cooperative Adaptive Cruise Control

Figure 4.10 – A vehicle’s reaction to an intended change in acceleration.

However, this equation only describes the change in acceleration from the moment the vehicle starts accelerating or decelerating. In reality, there is a significant delay between the moment a vehicle has the intention to brake (for instance, at the moment when the brakes are hit) and the moment the vehicle actually starts decelerating. We call this actuation delay $\theta$. This means that the reaction of a vehicle to the intended change in acceleration can be described by:

$$ a_c(t) = \begin{cases} 
  a_c(0) & \text{if } t < \theta_c \\
  a_c(0) + (1 - e^{-(t-\theta_c)/\tau_c})(a_{c,\text{int}} - a_c(0)) & \text{if } t \geq \theta_c 
\end{cases} \tag{4.5} $$

This equation is plotted in Figure 4.10. In this example, we took $\theta_c = 0.15s$, $\tau_c = 0.1s$, $a_c(0) = 0m/s^2$ and $a_{c,\text{int}} = -10m/s^2$. These values are taken from Ploeg et al. (2011a).

Using equation 4.5, we are able to define a set of equations that ultimately calculates the difference in displacement between two vehicles given an emergency braking action at $t = 0$. For the sake of readability, we ignore the initial delay $\theta$ in deriving the time until standstill. In the following derivations of the vehicles' displacement, we focus on the period of actual braking and disregard the initial delay $\theta$; it will be reinserted in the final calculations. Thus, in the derivations $t = 0$ actually means the start of braking proper at $t = \theta$.
Chapter 4. Safety in the Face of Uncertainty

Preceding Vehicle In the emergency braking action that we model, the preceding vehicle’s displacement is the same for all three controller cases: the vehicle brakes with an intended acceleration \(a_{c,\text{int}} = -10\,\text{m/s}^2\). Using equation 4.5, we can calculate the preceding vehicle’s displacement \(x_p\).

When a force is applied to a braking pedal and this force represents a certain deceleration, this value shall not be instantaneously met, since the vehicle dynamics does not allow such behaviour. So let’s start with a model for the acceleration of the preceding vehicle \(p\), when maximal braking of \(a_{p,\text{max}}\) is applied while the vehicle has a current acceleration of \(a_p(0)\). In this case the desired acceleration equals \(a_{p,\text{max}}\):

\[
a_p(t) = a_p(0) + (1 - e^{-t/\tau_p}) (a_{p,\text{max}} - a_p(0))
\]  

(4.6)

We picked the reasonable value for time constant \(\tau\) equals 0.1s. This value is also assumed for the preceding vehicle. When the preceding vehicle \(v_p\) brakes with the above acceleration function the velocity of \(v_p\) during this braking action can be derived by integrating equation 4.6:

\[
v_p(t) = \int_0^t a_p(t) \, dt \\
= a_p(0) t + (t + \tau_p e^{-t/\tau_p}) (a_{p,\text{max}} - a_p(0)) \\
- \tau_p (a_{p,\text{max}} - a_p(0)) + v_p(t_{\text{brake}})
\]

(4.7)

with \(v_p(t_{\text{brake}})\) the velocity of \(v_p\) when the deceleration starts, not at \(t = 0\). This is because the velocity of the vehicle may change during the actuator delay \(\theta_p\), because \(a_{p,\text{i0}}\) may be non-zero. We assume that during this delay, the acceleration stays constant. Thus, \(v_p(t_{\text{brake}})\) is given by:

\[
v_p(t_{\text{brake}}) = v_p(0) + a_p(0) \theta_p
\]

(4.8)

The amount of time it takes the preceding vehicle to come to a standstill \(T_p\) can be calculated by substituting \(t\) with \(T_p\) in equation 4.7 and solving it:

\[
a_p(0) T_p + (T_p + \tau_p e^{-T_p/\tau_p}) (a_{p,\text{max}} - a_p(0)) \\
- \tau_p (a_{p,\text{max}} - a_p(0)) + v_p(t_{\text{brake}}) = 0
\]

(4.9)
4.3. Critical Headway Estimation in Cooperative Adaptive Cruise Control

This can be solved when dropping the term $\tau_p e^{-T_p/\tau_p}$; we can safely do this because this term is negligible for $\tau_p = 0.1$. Now we can derive $T_p$:

$$T_p = \frac{\tau_p (a_{p,\text{max}} - a_p(0)) - v_p(t_{\text{brake}})}{a_{p,\text{max}}}$$

Substitution of $v_p(t_{\text{brake}})$ then yields

$$T_p = \frac{\tau_p (a_{p,\text{max}} - a_p(0)) - (v_p(0) + a_p(0)\theta_p)}{a_{p,\text{max}}}$$

Now the traveled distance (braking path) of the preceding vehicle can be calculated, while taking into account its actuation delay $\theta_p$:

$$x_{p,\text{total}} = x_{p,\text{init}} + x_{p,\text{brake}}$$

$$= \int_0^{\theta_p} \int_0^t a_p(0) dt \, d\bar{t} + \int_{\theta_p}^{\theta_p + T_p} v_p(t - \theta_p) \, dt$$

$$= \int_0^{\theta_p} \int_0^t a_p(0) dt \, d\bar{t} + \int_{0}^{T_p} v_p(t) \, dt$$

$$= \frac{1}{2} a_p(0) \theta_p^2 + v_p(0) \theta_p + \int_0^{T_p} a_p(0) t + (t + \tau_p e^{-t/\tau_p})(a_{p,\text{max}} - a_p(0))$$

$$- \tau_p (a_{p,\text{max}} - a_p(0)) + v_p(t_{\text{brake}}) \, dt$$

$$\approx \frac{1}{2} a_p(0) \theta_p^2 + v_p(0) \theta_p + \frac{1}{2} a_{p,\text{max}} T_p^2$$

Substitution of $v_p(t_{\text{brake}})$ yields:
\[
x_{p,\text{total}} = \frac{1}{2}a_p(0)\theta_p^2 + v_p(0)\theta_p + \frac{1}{2}a_{p,\text{max}}T_p^2 \\
- \tau_p(T_p - \tau_p)(a_{p,\text{max}} - a_p(0)) \\
+ (v_p(0) + a_p(0)\theta_p)T_p
\]

It is important to note that the above equations hold for the preceding vehicle in general, regardless of the safety checker mode that is used in the host vehicle. However, in the eACC case, the host vehicle does not know the value of \(a_{p,0}\) directly, because only in the CACC cases, the acceleration value is directly transmitted. Thus, we need a model for determining the preceding vehicle’s acceleration. We use a simple model (that could easily be substituted with a more complex one when available) where the vehicle estimates this value from two consecutive radar measurements of the velocity. This means that for the eACC case, in the above equations each occurrence of \(a_p(0)\) must be substituted with:

\[
a_p(0) = \frac{v_p(0) - v_p(-1)}{\Delta t}
\]

with \(v_p(0)\) the current velocity of the preceding vehicle, \(v_p(-1)\) the last measured velocity of the preceding vehicle, and \(\Delta t\) the time between two measurements. With a radar that measures at 10Hz, \(\Delta t\) equals 0.1s.

**Host vehicle using the eACC mode** The host vehicle also works according to the vehicle model, as depicted in equation 4.5. However, instead of having an own intended acceleration \(a_{h,\text{int}}\), the vehicle bases its own acceleration on the acceleration of the preceding vehicle.

The deceleration of the host vehicle can now be given as a function of the deceleration curve of the preceding vehicle:

\[
a_h(t) = a_h(0) + (1 - e^{-t/\tau_h})(a_p(t) - a_h(0))
\]

Substitution of \(a_p(t)\) with equation 4.6 yields:
4.3. Critical Headway Estimation in Cooperative Adaptive Cruise Control

\[
a_h(t) = a_h(0) + (1 - e^{-t/\tau_h})((a_p(0) + (1 - e^{-t/\tau_p})(a_{p,\text{max}} - a_p(0))) - a_h(0)) \\
= a_p(0) + \left(1 - e^{-t/\tau_h} - e^{-t/\tau_p} + e^{-t(\tau_{h}+\tau_{p})/\tau_{h}^{p}}\right) \\
\cdot (a_{p,\text{max}} - a_p(0)) + e^{-t/\tau_h}(a_h(0) - a_p(0))
\]

(4.16)

Note that in equation 4.16, \( t = 0 \) is the time at which the vehicle starts decelerating. This means that in the formulas for the displacement, the actuator delays of both the host and the preceding vehicle still need to be taken into account.

Remember that, as explained above, in all following equations for eACC, each occurrence of \( a_p(0) \) must be substituted with equation 4.14. Since this clutters up our equations a lot, we denote this substitution with \( A_p(0) \). We get:

\[
a_h(t) = A_p(0) + \left(1 - e^{-t/\tau_h} - e^{-t/\tau_p} + e^{-t(\tau_{h}+\tau_{p})/\tau_{h}^{p}}\right) \\
\cdot (a_{p,\text{max}} - A_p(0)) + e^{-t/\tau_h}(a_h(0) - A_p(0))
\]

(4.17)

We integrate this equation to get the formula for the velocity of the host vehicle:

\[
v_h(t) = \int_0^t a_h(t)dt \\
= A_p(0)t + \left(t + \tau_h e^{-t/\tau_h} + \tau_p e^{-t/\tau_p} - \frac{\tau_p \tau_h}{\tau_p + \tau_h} e^{-t(\tau_{h}+\tau_{p})/\tau_{h}^{p}}\right)(a_{p,\text{max}} - A_p(0)) \\
+ \tau_h e^{-t/\tau_h}(a_h(0) - A_p(0)) \\
- \frac{\tau_p \tau_h}{\tau_p + \tau_h} \tau_{h}(a_{p,\text{max}} - A_p(0)) \\
- \tau_h(a_h(0) - A_p(0)) + v_h(t_{\text{brake}})
\]

(4.18)

Using the same approach as for the preceding vehicle, we can substitute \( t \) with \( T_h \), drop the terms with \( e^{-\cdot} \) because they are negligible for small values of \( \tau \), and solve it to derive the equation for \( T_h \):
Chapter 4. Safety in the Face of Uncertainty

\[ T_h = \frac{\left( \tau_h + \tau_p - \frac{\tau_p \tau_h}{\tau_p + \tau_h} \right) (a_{p,\text{max}} - A_p(0))}{a_{p,\text{max}}} + \frac{\tau_h (a_h(0) - A_p(0)) - (v_h(0) + a_h(0)(\theta_p + \theta_h))}{a_{p,\text{max}}} \]  

(4.19)

Now we can derive the formula for displacement of the eACC host vehicle by integrating equation 4.18. We also have to add a term for the initial displacement during the actuator delays of both vehicles.

\[ x_{h,\text{total}} = x_{h,\text{init}} + x_{h,\text{brake}} \]

\[ = \int_0^{\theta_h + \theta_p} \int_0^t a_h(0) dt \, d\tilde{t} + \int_0^{\theta_h + \theta_p + T_h} v_h(t - (\theta_h + \theta_p)) dt \]

\[ = \int_0^{\theta_h + \theta_p} \int_0^t a_h(0) dt \, d\tilde{t} + \int_0^{T_h} v_h(t) dt \]

\[ = \frac{1}{2} A_p(0)(\theta_h + \theta_p)^2 + v_h(0)(\theta_h + \theta_p) + \frac{1}{2} A_p(0) T_h^2 \]

(4.20)

\[ + \left( \frac{1}{2} T_h^2 - T_h (\tau_h + \tau_p - \frac{\tau_h \tau_p}{\tau_h + \tau_p}) \right) (a_{p,\text{max}} - A_p(0)) \]

\[ - \tau_h T_h (a_h(0) - A_p(0)) + (v_h(0) + a_h(0)(\theta_h + \theta_p)) T_h \]

\[ + \left( \frac{T_h^2}{2} + \frac{T_p^2}{2} - \left( \frac{\tau_p \tau_h}{\tau_p + \tau_h} \right)^2 \right) (a_{p,\text{max}} - A_p(0)) \]

\[ + \tau_h^2 (a_h(0) - A_p(0)) \]

The difference in displacement is now simply:

\[ \Delta s = x_{h,\text{total}} - x_{p,\text{total}} \]  

(4.21)

Note that the radar processing time and the radar frequency implications are not yet included in this model. We include these aspects in Section 4.3.4.

Host vehicle using eCACC₁ The eCACC₁ safety checker is very similar to the ACC safety checker. The big difference is that the acceleration of the preceding vehicle is no longer estimated by the radar, but directly communicated by the
preceding vehicle. This means that we can use \( a_{p,t0} \) directly in our equations this time. This means that we have the same equations as in the ACC case, except now we substitute \( A_{p,t0} \) with \( a_{p,t0} \) in the equation for \( x_{h,total} \).

The uncertain parameters have slightly different values in this setting. Also, in this setting we have a communication delay. We come back to these aspects thoroughly in our validation section.

**Host vehicle using eCACC\(_2\)** The most innovative controller has the simplest set of equations. The reason for this is that the eCACC\(_2\) controller is able to communicate its intended acceleration to the following vehicle. This means that the following vehicle can, in case of emergency, copy this desired acceleration. Therefore, the host vehicle can set its own intended acceleration to the same value of the preceding vehicle, before the preceding vehicle even starts decelerating. This makes the host vehicle much more responsive than the other controllers.

When the host vehicle has the same dynamics as its preceding vehicle, our equations become rather simple: the host vehicle will behave along the same curve as the preceding vehicle. The equation for difference in displacement of the vehicles is:

\[
\Delta s = x_{h,\text{total}} - x_{p,\text{total}} = \frac{1}{2} a_h(0)\theta_h^2 + v_h(0)\theta_h + \frac{1}{2} a_{h,\text{max}} T_h^2
- \tau_h (T_h - \tau_h) (a_{h,\text{max}} - a_h(0))
+ (v_h(0) + a_h(0)\theta_h) T_h
- \frac{1}{2} a_p(0)\theta_p^2 - v_p(0)\theta_p - \frac{1}{2} a_{p,\text{max}} T_p^2
+ \tau_p (T_p - \tau_p) (a_{p,\text{max}} - a_p(0)) - (v_p(0) + a_p(0)\theta_p) T_p
\]

with

\[
T_h = \frac{\tau_h (a_{h,\text{max}} - a_h(0)) - (v_h(0) + a_h(0)\theta_h)}{a_{h,\text{max}}} \quad (4.23)
\]

Note that in equation 4.23, only the actuator delay of the host vehicle is included. The actuator delay for the preceding vehicle can be omitted because the host vehicle can directly react on the intended acceleration. In the descriptions of
the controllers above, we have omitted the delay of communication and the radar processing delay. The reason for this is that these delays work differently than the actuator delays. When a radar reading is processed, or when a communication message is received, the information that is contained within is delayed. The delays are uniformly distributed between 0s and 0.1s (if the frequency is 10Hz). To deal with this issue, we have to perform the convolution operation on the results. We explain this in more detail in the experiments section.

Both e-CACC cases use time-based communication as opposed to event-based communication. On the one hand, this makes our method more robust, since receiving vehicles know when to expect new messages and are able to anticipate when a message does not arrive. This is not the case when event-based communication is used. On the other hand, time-based communication is less scalable. When many vehicles within a certain range are constantly broadcasting messages, this will influence the quality of the network.

4.3.3.2 Uncertainty & Delay

The equations that we described in the previous sections are fairly straightforward, and if the values of all variables are known, one can easily derive the difference in displacement of the vehicles, and after adding communication and radar processing delays, one could compute the distance at which the vehicles would not crash. However, since sensor information and wireless communication comes with a lot of uncertainty, the computation of the critical safe headway time is not a trivial task. In this section, we describe the various factors of uncertainty that are present in our sensors.

The following variables that we use in the equations that we introduced in the previous section are uncertain:

- Host’s and preceding vehicle’s velocity \(v_h(0)\) and \(v_p(0)\);
- Host’s and preceding vehicle’s acceleration \(a_h(0)\) and \(a_p(0)\);
- Host’s and preceding vehicle’s maximum deceleration \(a_{h,max}\) and \(a_{p,max}\);
- Host’s and preceding vehicle’s actuator delay \(\theta_h\) and \(\theta_p\);
- Host’s and preceding vehicle’s vehicle model time constant \(\tau_h\) and \(\tau_p\);
- Uncertainty in radar range measurement \(\sigma_r\).
In the analytical method, we calculate the resulting difference in displacement $\Delta s$ once by using the equations in section 4.3.3. The difference in displacement is used as the mean in the resulting distribution that we construct.

Then, we make use of the following well-known theorem to combine the normally distributed variables in order to obtain the standard deviation of $\Delta s$: if $X_i$ are independent random variables and $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ for $i = 1, ..., n$ then:

$$\sum_{i=1}^{n} a_i X_i \sim \mathcal{N}\left( \sum_{i=1}^{n} a_i \mu_i, \sum_{i=1}^{n} (a_i \sigma_i)^2 \right)$$ (4.24)

Under this assumption, we can now calculate the uncertainty of the difference in displacement of the vehicles. To this end, we need to linearise the equation for $\Delta s$ to the values of each uncertain variable in it, after which we can apply theorem 4.24 to calculate the standard deviation of $\Delta s$:

$$\sigma_{\Delta s} = \sqrt{\sum_{i}^{n} \left( \frac{\partial \Delta s}{\partial X_i} \right)^2 \sigma_i^2}$$ (4.25)

This means that we differentiated $\Delta s$ to each random variable. We have not included these derivatives in this section, but they are available from the author’s website\(^{ii}\).

Now we have a mean and a standard deviation, which we use to construct a probability density function for the difference in displacement. We then include the communication delays and/or radar processing delays, depending on the capabilities of the vehicle. There are two types of operation: shifts and convolutions. We shift the function to the right in case of simple latency. For both radar and communication, we have 10ms latency, so the function of all controllers are shifted to the right by the equivalence of 10ms in distance (in our case, 0.3m).

A convolution must now be done in order to deal with the delay of the information contained in the communication messages or the radar readings. The information in these lie uniformly somewhere between 0 and 0.1s. Therefore, a convolution with a boxcar function with the interval [0,0.1] is performed.

\(^{ii}\)http://www.researchgate.net/profile/WH_Van_Willigen
Then, we take the $3\sigma$ tail of the resulting probability density function as the critical headway distance $\Delta s_{\text{crit}}$. This safe headway distance can be converted to safe headway time $h_{\text{crit}}$ by dividing it with the velocity:

$$h_{\text{crit}} = \frac{\Delta s_{\text{crit}}}{v}$$  \hspace{1cm} (4.26)

This method and these analytical expressions can be used with any vehicle model, and any model of uncertainty, as long as these models are known. In the next section, we show how our method can be used to determine critical safe headway times, given some instantiation of our uncertain variables.

The equations that were described above take into account two consecutive vehicles. When more vehicles are present in the platoon, the same equations hold for each pair of vehicles in the platoon, as each vehicle acts on observations of the preceding vehicle. This means that the method does not degrade when more vehicles are present in the platoon.

### 4.3.4 Validation

In this section, we describe how our method can be used. We instantiate our method with certain parameters, and show how the critical headway time can be calculated, given the values of these parameters. Furthermore, we validate the results of our calculations by means of Monte Carlo simulations.

#### 4.3.4.1 Scenario

The objective of our method is to find a critical safe headway time for different adaptive cruise control controllers fast, given various parameters that describe factors of uncertainty in the system.

To this end, we imagine two vehicles on a highway, of which the front vehicle applies an emergency brake at $t = 0$. The following vehicle has to respond to this emergency brake. The way the following vehicle responds to the emergency brake, depends on the capabilities of the vehicle. To calculate the difference in displacement of the vehicles, we use the equations as described in section 4.3.3. As explained, if the vehicle has only radar, it uses the $e\text{ACC}$ checker, the $e\text{CACC}_1$ mode uses direct communication of the actual deceleration, and the $e\text{CACC}_2$ mode uses direct communication of the predicted deceleration. In Figures ?? and ??, these scenarios are illustrated.
### 4.3. Critical Headway Estimation in Cooperative Adaptive Cruise Control

<table>
<thead>
<tr>
<th>Param</th>
<th>$\mu$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_p(0)$</td>
<td>30</td>
<td>0.2</td>
</tr>
<tr>
<td>$a_p(0)$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$a_{p,\text{max}}$</td>
<td>-10</td>
<td>0.2</td>
</tr>
<tr>
<td>$\theta_p$</td>
<td>0.15</td>
<td>0.01</td>
</tr>
<tr>
<td>$\tau_p$</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td>$v_h(0)$</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>$a_h(0)$</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>$a_{h,\text{max}}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\theta_h$</td>
<td>0.15</td>
<td>0.01</td>
</tr>
<tr>
<td>$\tau_h$</td>
<td>0.1</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 4.4 – Values of each uncertain parameter per controller in our experimental setup. If no value is reported, that variable is not relevant for that particular controller.
Chapter 4. Safety in the Face of Uncertainty

(a) The eACC scenario, in which the host vehicle uses its radar to obtain information about velocity from its preceding vehicle, which is used to derive information about acceleration.

(b) The e-CACC scenario, in which the preceding vehicle uses direct communication to transmit information about its acceleration to the host vehicle.

Figure 4.11 – The difference between the ACC and CACC scenario.

4.3.4.2 Instantiating our method

The values of the means and standard deviations of each of the uncertain parameters that we introduced in section 4.3.3.2 are summarised in table 4.4. These values can be interpreted as the situation on the road at some given moment. The values in this table are continuously updated by the sensors as the vehicle drives on the road and as the conditions change.

Furthermore, we introduce the following fixed delays and frequencies:

- Radar processing delay: 50ms. This is how long it takes before a radar signal is processed and ready to use;

- Communication delay: 10ms. This is how long it takes for a message to be received and processed;

- Communication frequency: 10Hz;

- Radar frequency: 10Hz.

Others include delays in bus, gateway and radio channel access. These are not included, but incorporation into this method is trivial, as long as the effect on $\Delta s$ is known.
4.3. Critical Headway Estimation in Cooperative Adaptive Cruise Control

**Table 4.5** – Critical headway distance & time for the three checkers using the analytical method.

<table>
<thead>
<tr>
<th></th>
<th>eACC</th>
<th>eCACC₁</th>
<th>eCACC₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta s_{\text{crit}}$</td>
<td>11.32</td>
<td>10.07</td>
<td>6.69</td>
</tr>
<tr>
<td>$h_{\text{crit}}$</td>
<td>0.38</td>
<td>0.34</td>
<td>0.22</td>
</tr>
</tbody>
</table>

**Table 4.6** – Critical headway distance & time for the three checkers using the Monte Carlo method.

<table>
<thead>
<tr>
<th></th>
<th>eACC</th>
<th>eCACC₁</th>
<th>eCACC₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta s_{\text{crit}}$</td>
<td>11.23</td>
<td>10.08</td>
<td>6.76</td>
</tr>
<tr>
<td>$h_{\text{crit}}$</td>
<td>0.37</td>
<td>0.34</td>
<td>0.23</td>
</tr>
</tbody>
</table>

When our method is instantiated, the critical safe headway time is calculated from the analytical expressions from section 4.3.3.

4.3.4.3 Results & Validation

The red lines in Figures 4.12, 4.13, and 4.14, show the probability density functions that we constructed from the mean and standard deviations that we obtained from our calculations. In Table 4.5, we show the $3\sigma$ values from these constructed functions. Note that these results are not fixed critical headway times that count in every situation on the road - rather, these are only the critical safe headway times given the parameter values that are summarised in Table 4.4. In practice, these values may change continuously.

To validate our method, we compared the results with a brute-force Monte Carlo method. In this approach, we sample the uncertain parameters many times (100,000 time per controller), and use the equations from section 4.3.3 to compute the difference in displacement. This results in a distribution of $\Delta s$, rather than an analytical expression for the distribution of $\Delta s$. In these distributions, we also include the communication delays and/or radar processing delays, in the same way we did this in this analytical method. The difference is that in this approach, we are working with distributions of simulated data for $\Delta s$, rather than constructed probability density functions.

In Figures 4.12, 4.13, and 4.14, the blue histograms show the results from our Monte Carlo validation runs. From these histograms, we determined the $3\sigma$ values. These are summarised in Table 4.6.
Chapter 4. Safety in the Face of Uncertainty

Figure 4.12 – Results for eACC

Figure 4.13 – Results for eCACC$_1$

Figure 4.14 – Results for eCACC$_2$
The analytical method obtains roughly the same results as the Monte Carlo method. The constructed probability density functions and the histograms of the Monte Carlo validation runs lie neatly on top of one another, and the resulting $3\sigma$ critical headway distances are very close to each other. Thus, we validated our analytical method using the Monte Carlo method, ensuring that the analytical method is accurate enough to be used as an on-line safety checker.

One limitation of our real-time safety checker is that it requires the random variables to be independent and normally distributed. If one or more of the variables are not normally distributed, it is no longer possible to provide an analytical expression for $\Delta s$. This can be solved by calculating the critical safe headway time given all normally distributed uncertain parameters, and perform a convolution with the not normally distributed parameter(s). The downside to this solution is that this convolution operation turns the analytical solution into a numerical one, which requires more computational resources. The time complexity of this solution is linear in the number of parameters that are not normally distributed, given that these parameters are uncorrelated.

A great benefit of our method is that it runs in real time. We profiled the non-compiled Matlab code for the eCACC$_2$ checker on a four-year-old computer, which resulted in run-times of just over 0.1 seconds. While this does not comply with our definition of real time (which would require the code to run in under 0.1 seconds), we can conclude that a compiled and further optimised version of the code would easily run in under 0.1 seconds on newer, dedicated hardware.

### 4.3.5 Conclusion

We have argued that it is crucial to consider current uncertainties in vehicle state, vehicle behaviour and communication to guarantee safety in intelligent vehicle control.

We presented an analytical method for approximating, in real time, the critical headway time between two vehicles in Adaptive Cruise Control settings that takes these uncertainties into account. Our method can reliably check whether the current headway time is safe, automatically reacting to abrupt changes in the leading vehicle’s speed, such that the safety can be guaranteed under these uncertain circumstances.
Chapter 4. Safety in the Face of Uncertainty

We elaborated our method for three different cases, corresponding to vehicles with different capabilities: eACC, that only uses radar; eCACC\(_1\), where the vehicles exchange information about the current velocity and acceleration; and eCACC\(_2\), in which the intended change in acceleration is communicated as well. We compared these cases with Monte Carlo simulations where the lead vehicle brakes abruptly and showed that the calculated critical headway times match with the ones from the Monte Carlo simulations.

The cases we showed all work under the assumption that the uncertainties are normally distributed. The uncertainty in the delay of communication is not normally but uniformly distributed and therefore was included by means of a convolution. In the case one or more of the other uncertainties are also not normally distributed, it can also be taken into account by means of a convolution.

In future work we will focus on implementing the proposed methods in real vehicles, dealing with real data, calculating the current critical headway time in real time.
Imagine a driver going from home to work. When she enters the highway, she turns on her auto-pilot, that completely takes over the control of the vehicle. She may want to read the newspaper or review a presentation she must give later that day. In this situation, driving comfortably has a clear preference over getting to work as quickly as possible. However, during the trip she gets a call that her first meeting is moved and suddenly she’s in a hurry. Now, getting there fast is the main objective and the driving style of the vehicle is adjusted accordingly.

This example shows that different circumstances lead to different preferences and that even in autonomous vehicles, drivers want their vehicle to drive in the way they prefer. We describe a method that uses an evolutionary algorithm to evolve not one, but multiple controllers, each having their own unique prioritisation of driver preferences. While this research doesn’t include the interface design, we imagine that this interface would consist of several sliders, each indicating how important the driver considers a certain preference.

This chapter describes an algorithm that uses several functions that represent driver objectives, such as speed and fuel economy, to evolve a set of autonomous vehicle controllers, each optimising on a unique prioritisation of these objectives.
The controllers of our autonomous vehicles are implemented as neural networks, that take sensor information as inputs and execute an action accordingly. These neural networks are evolved using a multi-objective version of the well-known NEAT algorithm (Stanley and Miikkulainen, 2002). We did extensive experiments with this algorithm and up to six different objectives, that represent driver preferences such a speed, comfort and fuel economy.

We show that it is in principle possible and beneficial to employ multi-objective evolution to develop useful sets of controllers as described.

Adding objectives can result in a slight degradation in terms of quality of controllers on originally considered objectives, although in some cases, performance on the original objectives increases when adding an objective.

It seems that substantial performance loss is not the result of considering multiple objectives per se (at least not up to five or six objectives), but that some combinations of objectives hamper the evolutionary process and result in suboptimal solutions.

The concept of evolving many non-dominated controllers represents a significant improvement over the current practice of optimising a weighted average of objectives to obtain a single controller that prioritises ride qualities as the developer deems appropriate. The consequent freedom of choice for drivers to select controller behaviour as they see fit will benefit acceptance of autonomous vehicle technology.

This chapter consists of four sections, and each section builds on the previous one. This means that much of these sections are quite repetitive, especially the introductory and background parts. Section 5.1 describes our first efforts, using two objectives (speed and effort). In section 5.2, we made some fundamental changes to the inputs and outputs of the neural network. We had the insight that we needed platooning-related inputs and outputs, which should make the neural network learn more efficiently. We also expanded the scope of the experiments in this section: we did many more repeats for each individual to have a more robust performance measure. In section 5.3, we expanded the experiments in terms of number of objectives: we did experiments with 6 different objectives, instead of only 2. And finally, in section 5.4, we did experiments with any possible combination of these 6 objectives.
5.1 Evolving Intelligent Vehicle Control using Multi-Objective NEAT

The research in this section is inspired by a vision of intelligent vehicles that autonomously move along motorways: they join and leave trains of vehicles (pla-toons), overtake other vehicles, etc. We propose a multi-objective algorithm based on NEAT and SPEA2 that evolves controllers for such intelligent vehicles. The algorithm yields a set of solutions that embody their own prioritisation of various user requirements such as speed, comfort or fuel economy. This contrasts with most current research into such controllers, where the user preferences are summarised in a single number that the controller development process should optimise. Having multiple prioritisations of preferences would, however, allow the user to select desired vehicle behaviour in real time, for instance fast driving if she’s in a hurry or economical driving in more relaxed circumstances. Preliminary results of our experiments show that evolved controllers substantially outperform the human behavioural model. We show that it is possible to evolve a set of vehicle controllers that correspond to different prioritisations of user preferences, giving the driver, on the road, the power to decide which preferences to emphasise.

5.1.1 Introduction

In Europe’s major cities, the average driver loses as much as 60 to 70 hours per year in traffic jams,¹ the cause of tremendous environmental and socio-economical impact. Vehicle manufacturers are introducing Adaptive Cruise Control (ACC) to help combat this issue. ACC allows the driver to set the desired distance to the

---


Section 5.1 was published as:

Chapter 5. Preference-based Controller Building

Figure 5.1 – A schematic representation of the difference between ACC and CACC.

preceding vehicle. The ACC system measures the vehicle’s relative distance and velocity through radar and adjusts speed accordingly. Such automated maintenance of distance is hoped to increase the uniformity of traffic flows and so better utilise road capacity.

Cooperative Adaptive Cruise Control (CACC) is a major development in recent research on Intelligent Transportation Systems (ITS) that takes this automated control to the next level by adding direct communication between vehicles (see Figure 5.1). This enables the vehicles to directly communicate about their current velocity, position and acceleration and so removes some of the uncertainty and delays that relying only on radar measurements implies. Directly communicating such accurate state information allows vehicles to drive much closer to each other without compromising safety. This opens up the possibility of platooning: trains of vehicles that drive very close together at (near) equal speed and therein lies one of the big selling points of CACC. It enables platoons to be string stable: changes in acceleration or deceleration are reduced by the following vehicles instead of amplified. This property is expected to greatly improve the throughput of vehicles on highways, because it is exactly the amplification of acceleration and deceleration that causes many traffic jams.

CACC will become available in the near future. Vehicles will be equipped with a simple button that turns the CACC mode on and off again. With the system
turned on, the vehicle will then, for instance, search for the nearest platoon available to join and (almost) automatically drive along in the platoon until the driver decides to take matters into her own hands and turn CACC off again. Such CACC systems do not yet take into account the preferences of specific drivers. Some drivers like to drive fast, while others prefer fuel economy and care less about speed. Ride comfort is also a consideration that can be measured in number of lane changes, changes in velocity, etc.

We envision a system where the driver can enter her preferences into the system at real time. Setting preferences corresponds with selecting a high-level vehicle controller. These controllers choose actions (e.g., joining, leaving or creating platoons, changing lanes or simply driving on) based on a number of input variables such as acceleration, velocity and distance to the preceding vehicle. This vision is at odds with most current research into controller development because of the way driver preferences are typically taken into account: the preferences are all summarised into one single metric (e.g. a weighted average) that is then optimised.

We propose to use multi-objective evolutionary algorithms as an approach to develop controllers without having to combine the separate preference metrics into a single number. This approach yields a number of Pareto-optimal controllers that each reflect their own prioritisation of preference measures. Some may be fast but less comfortable, others may be very comfortable but slow, yet others may strike another balance between the user preferences. Such a selection of controllers would enable a user to set the relative importance of fuel economy, speed, lane changes, etc. The controller that best approximates the current prioritisation of preferences is then activated and vehicle progresses as the user has requested. Thus, we seek the benefit of evolutionary multi-objective optimisation of autonomous vehicle controllers not so much in better control (although that would of course not be disregarded), but more in the fact that it would yield multiple controllers with varying emphasis on the different user preferences and thus allow the user to select the appropriate vehicle behaviour in real time.

The method we propose employs NEAT, a well-known neuro-evolutionary algorithm (Stanley and Miikkulainen, 2002). We combine NEAT with techniques for multi-objective optimisation from the SPEA2 evolutionary algorithm (Zitzler et al., 2002), which results in NEAT-SPEA2, where we effectively ‘multi-objectified’ the NEAT algorithm.
As a proof-of-concept, we conduct a number of simulated experiments in which we evolve controllers to optimise both speed and comfort as twin objectives. Our first research question is whether our NEAT-SPEA2 approach does indeed yield a useful set of controllers that embody different prioritisations of these objectives. Secondly, we compare the performance of the evolved controllers with a benchmark, the Intelligent Driver Model (IDM) (Treiber et al., 2000) in combination with the MOBIL lane changing model (Treiber and Helbing, 2002), that are commonly used to model human drivers in traffic flow modelling.

5.1.2 Related Work

Our work is positioned within Cooperative Adaptive Cruise Control (CACC) systems, specifically with platooning as a possible application. While ACC systems are currently being adopted in consumer vehicles, research and development into cruise control focuses on enabling more and better cooperation between ACC systems, yielding these CACC systems. Van Arem et al. (2006) describe the effect of CACC on traffic flow. They conclude that, when the penetration level of CACC-equipped vehicles is high enough (> 60%), traffic stability and throughput is improved. In Yang et al. (2004), a communication protocol is proposed in order to make a cooperative collision warning system on highways.

The main application area of CACC technology these days is platooning. Broggi et al. (2000) and Kanellopoulos et al. (1999) both use image recognition techniques in combination with sensors to autonomously enable platooning. However, current technology has improved significantly since then and nowadays direct radio communication between vehicles is used to enable platooning.

Naus et al. (2010) thoroughly investigate the issue of string stability in platooning, with both ACC and CACC controllers. Their method includes several factors of delay in communication.

In Hallé (2005), an extensive architecture is given for a layered multi-agent CACC architecture. The authors use this architecture to implement both centralised platoons (in which there is a coordinating platoon leader) and decentralised platoons (in which all vehicles operate as equals). Khan et al. (2008) present different platoon (in their paper, convoy) forming strategies, based on a utility value of a platoon.
There has not been much work yet on the development of platooning strategies, where the vehicles have to decide while driving if and when they should be platooning. In the SARTRE project (Safe Road Trains for the Environment) (Bergenhem et al., 2010), platoons are defined as trains of road vehicles of which the front one is a trained platoon driver. They have implemented basic platooning strategies, also in real vehicles, but these strategies are only defined at the action level, where for example the join and leave actions are defined, but not at a higher level, where a strategy for driving on the highway is defined.

The multi-objective driving strategies domain has been investigated by Dovgan et al. (2011, 2012). They use the NSGA-II multi-objective evolutionary algorithm to determine driving strategies that optimise travel time and fuel economy. Other vehicles are not taken into account in this work, only the vehicle’s own state space and route state space. Platooning is not an application that is taken into account.

In the domain of human driver modelling, several well-known models exist, such as the simple yet elegant Intelligent Driver Model (Treiben et al., 2000) and the older Gipps model (Gipps, 1981).

Neuro-evolution is a form of machine learning that uses evolutionary algorithms to train artificial neural networks. There are many methods for neuro-evolution, most notably NEAT (Stanley and Miikkulainen, 2002) and HyperNEAT (Gauci and Stanley, 2007), an extension to NEAT, which uses an extra layer of encoding to represent higher-level pattern structures such as symmetry and repetition.

There are many multi-objective evolutionary algorithms, among which two famous ones are SPEA2 (Zitzler et al., 2002) and NSGA-II (Deb et al., 2002). SPEA2 calculates a single fitness value based on the number of other individuals an individual dominates, which is then used in the evolutionary process. NSGA-II uses another mechanism, based on the rank of an individual in the population, which also uses the domination relation, but doesn’t attach a specific fitness value to it.

In summary, our work touches upon many different domains of research, including Intelligent Transportation Systems and platooning, multi-objective optimisation and evolutionary computing. It is an interesting and challenging task to combine techniques from these different research areas.
5.1.3 Algorithm

Our experimental set-up consists of two main parts: the vehicle simulator in which we tested our controllers and the multi-objective NEAT algorithm that evolves the vehicle controllers based on the feedback it obtains from the simulator. This setup is schematically depicted in Figure 5.2.

![Figure 5.2](image)

**Figure 5.2** – Graphical representation of our simulation loop. The NEAT algorithm generates a new population of controllers. Each member of a new generation of vehicle controllers is then evaluated by the vehicle simulation. The simulator assigns the fitness values to the controllers. Then, until a fixed number of generations has been generated, NEAT creates a new generation based on the evaluated individuals.

5.1.3.1 Vehicle Simulation

We used Movsim (Kesting, 2008), an existing, open-source vehicle simulator that implements the Intelligent Driver Model (IDM) and MOBIL, which are respectively the longitudinal model and the lane changing model that we used as a benchmark in our experiments. Movsim is primarily a microscopic simulator of traffic, whereas we want to use higher-level actions as input for the vehicle (such as creating, joining and leaving a platoon). Therefore we had to extend the functionality of the simulator to provide these higher-level actions. In the original simulator, each vehicle can only use a single longitudinal controller and this controller decides on every timestep what the acceleration of the vehicle should be.

In the extended simulator, each vehicle can use many controllers and a decision making process (such as a neural network) on a more abstract level selects the next action of the vehicle based on the current state of the vehicle. To execute
the selected action the vehicle decides on the appropriate longitudinal controller for that moment. For example, when joining a platoon, the vehicle first has to drive towards the platoon using a specific longitudinal controller that facilitates this. Then, when the vehicle has approached the platoon, it may have to change to the correct lane. Other vehicles may have to make space for the joining vehicle. All these processes correspond with different longitudinal controllers. In Figure 5.3, the extra functionality of the simulator is schematically shown. In the original situation, only the top and bottom layers exist; we extended Movsim with three extra layers in the middle to facilitate behaviours on a more abstract level.

![Diagram of five-layered architecture of vehicle controllers](image)

**Figure 5.3** – The five-layered architecture of our vehicle controllers. In the original simulation settings, the three layers in the middle are missing, thus calculating the acceleration directly based on the input variables.

We implemented realistic platooning behaviour, where vehicles on the highway drive very closely to their preceding vehicles while copying the preceding vehicle’s acceleration. The intricacies of the platooning control, such as sensing, information fusing and communication models are not taken into account in this simulation, since these issues should be solved at the lower levels of the system. They are not within the scope of this work and we assume these technologies to be working correctly and without error.

### 5.1.3.2 Platooning

By default, the vehicles in our simulation drive according to the Intelligent Driver Model (IDM) and the MOBIL lane changing model. Additionally, our vehicles can execute actions, each triggering other versions of longitudinal controllers.
Create Platoon If the vehicle is currently not part of a platoon, it creates a new platoon. The IDM longitudinal model remains active and the lane changing model is turned off. In our simulation, platoons cannot change lanes. The vehicle that creates the platoon automatically becomes the platoon leader—the foremost vehicle in the platoon.

Join Platoon If the vehicle is currently not part of a platoon, it tries to join the nearest platoon. This is only possible if there are no other vehicles in the way. If the vehicle drives next to the platoon it decides to join, the vehicles in the platoon make space to allow the vehicle to join. The longitudinal model is CACC, where the velocity and acceleration of the vehicle is directly copied from the preceding vehicle. If the headway time to the preceding vehicle of the platoon is higher than necessary, the longitudinal model is temporarily switched to the so-called Approacher longitudinal model, where the vehicle’s acceleration is slightly higher than the preceding vehicle’s acceleration.

Leave Platoon If the vehicle is part of a platoon, the vehicle leaves the platoon and starts driving using the IDM and MOBIL models again. Immediately, more headway time is created. Additionally, if the vehicle that leaves the platoon was driving somewhere in the middle of the platoon, the platoon is split into two parts. If the leaving vehicle was platoon leader, this role is transferred to the second vehicle in the platoon. If the leaving vehicle was the only vehicle in the platoon, the platoon is disbanded.

Change Lane If possible, the vehicle changes lanes. Our simulation currently consists of two lanes, so this action does not need a parameter telling which lane to change to.

Do Nothing The vehicle longitudinal controller remains unchanged.

5.1.3.3 Multi-Objective NEAT

NEAT (Stanley and Miikkulainen, 2002) is an algorithm for evolving neural networks, including their topologies. This means that we do not have to decide on a topology for the neural network beforehand, which is convenient and it also allows for the algorithm to come up with functionally different neural networks. It does this using a clever method of speciation of the neural network topologies, taking into account specific sub-structures within the topology. The neural
networks compete primarily with individuals within their own niches, instead of with individuals of the entire population. This technique makes it easier to protect innovations within the topologies.

Our experiments take multiple objectives into account. Currently, we have implemented two objectives: travel time and comfort. Since our objectives are somewhat incompatible, we want to obtain a Pareto front of controllers. Some controllers are faster but are less comfortable, other controllers are slower but much more comfortable. Once we have obtained this Pareto front of controllers, we can let the driver choose which preferences he/she wants to use in the vehicle at that time. Some users prefer speed, others prefer comfort. By letting the driver choose between objectives while driving, we allow for changes in driving behaviour to match the preferences of the person behind the wheel. The original NEAT algorithm does not use multiple objectives. Therefore, we extend the NEAT algorithm to allow for multi-objective optimisation.

We took the multi-objective fitness evaluation from the SPEA2 evolutionary algorithm and implemented this into NEAT. The advantage of this evaluation is that it computes a single fitness value for each individual, based on the values of all objective functions, making it rather straightforward to augment the NEAT algorithm with multi-objective capabilities.

The basis of the SPEA2 multi-objective evaluation is the domination relation:

\[
i \succ j \iff \forall o \in O : o_i \succeq o_j \wedge i \neq j\tag{5.1}\]

In this formula, the \(\succeq\) relation between objectives only specifies that objective \(o_i\) is equal or better than objective \(o_j\), without saying whether the objective is maximised or minimised. In our specific experiments, we are minimising both objectives.

After calculating each objective function’s value for all individuals, we calculate the Pareto Strength values for each individual \(i\) in population \(P\) as described in Zitzler et al. (2002):

\[
S(i) = |\{j|j \in P \wedge i \succ j\}|\tag{5.2}
\]

where \(| \cdot |\) stands for the cardinality of a set and \(\succ\) for the dominance relation. Now we can calculate the raw fitness values of each individual \(i\) by summing the strengths of the individuals \(j\) that dominate \(i\):
The final fitness value for an individual that we feed to NEAT is then given by:

\[
F(i) = \frac{10000}{1 + R(i)}
\]  

(5.4)

From these formulae, it follows that for each non-dominated individual, the raw fitness value is 0 and the final fitness has the maximum value of 10,000.

Since this fitness evaluation creates a single fitness value based on the multiple fitness metrics, we could easily plug this into the NEAT algorithm\textsuperscript{ii}.

### 5.1.3.4 Evolution Objectives

The experiments described in this section serve primarily as a proof-of-concept. In this study, we make use of the following two objectives: As a measure of the comfort of a ride, we take the number of times a vehicle changes lanes during a ride, assuming that less lane changes means a more comfortable ride. Our second objective in this study is speed, simply measured as the time it takes a vehicle to complete the stretch of motorway in our simulation. The experiments described in this study serve primarily as a proof-of-concept and therefore we have only used two objectives. These two objectives are easily extended by more objectives. Examples of such extra objectives that could be included are fuel economy (when a vehicle is platooning, it consumes less fuel than when it’s not) or other measures for comfort (for example, the so-called vehicle ‘jerk’, the variability of speed of the vehicle).

### 5.1.4 Experiments

In order for the NEAT algorithm to do its job, we need to define the inputs and the outputs of the algorithm. Schematically, these inputs and outputs are depicted in Figure 5.6. We define the following input variables for a vehicle \( v \):

- Headway time of vehicle to preceding vehicle: The distance to the preceding vehicle divided by the velocity of \( v \);

\textsuperscript{ii}We used the NEAT4j implementation, see http://neat4j.sourceforge.net
5.1. Evolving Intelligent Vehicle Control using Multi-Objective NEAT

- Fraction of desired velocity: The current velocity of $v$ divided by its desired velocity;

- Acceleration: The current acceleration of $v$;

- isPlatooning?: The current platooning status of $v$;

- Headway time to preceding vehicle on other lane: the distance to the first preceding vehicle on the other lane divided by $v$’s velocity.

These input variables can easily be altered or extended with other variables, but it has to be kept in mind that the vehicle should be able to register these variables autonomously and on a real-time basis. The values of the input variables are normalised to $[0, 1]$ (except the desired speed fraction, that can get a value higher than 1) in order for them to be more suitable inputs for the neural network.

The output variables are simply the different actions that we have defined in our simulation. The simulation picks the action with the highest corresponding output value.

To evaluate a neural network controller, we simulate a 10 km, two-lane stretch of road with 3,000 vehicles running that particular neural network entering per hour. Each vehicle has a desired speed of 30m/s, which is slightly randomised when a vehicle enters the simulation.

A vehicle running a neural network evaluates the neural network every 5 seconds and executes the action that the neural network proposes. The vehicle only executes the action if it is possible. For example, the vehicle will not change lanes if this would result in a crash. If an action is not executable, the vehicle will simply remain using the longitudinal controller it was already using, waiting another 5 seconds before re-evaluating the neural network.

We run this simulation for 10 minutes of simulation time (i.e. ca. 500 vehicles enter the stretch of road), after which we measure the mean number of lane changes of the vehicles and the mean travel times of the vehicles that successfully finished the 10km. These measurements are then used to calculate the controller’s Pareto strength.

We used a population size of 50 and run the cycle as depicted in Figure 5.2 for 50 generations. This whole process was repeated 30 times. From the 30 final generations that obtained from these runs, we distilled all Pareto optimal solutions.
Table 5.1 – Experimental Setup

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road length</td>
<td>10 km</td>
</tr>
<tr>
<td>Vehicle inflow</td>
<td>3,000 veh/hr</td>
</tr>
<tr>
<td>Desired speed</td>
<td>±30 m/s</td>
</tr>
<tr>
<td>Max deceleration</td>
<td>-3 m/s²</td>
</tr>
<tr>
<td>Simulation time</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Timestep size</td>
<td>0.20 seconds</td>
</tr>
<tr>
<td>Neural Network Evaluations</td>
<td>Every 5 seconds</td>
</tr>
<tr>
<td>Neural Network inputs</td>
<td>5</td>
</tr>
<tr>
<td>Neural Network outputs</td>
<td>5</td>
</tr>
<tr>
<td>Population Size</td>
<td>50</td>
</tr>
<tr>
<td>Generations</td>
<td>50</td>
</tr>
<tr>
<td>Repeats</td>
<td>30</td>
</tr>
<tr>
<td>Benchmark runs</td>
<td>100</td>
</tr>
<tr>
<td>Validation runs</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 5.4 – An example of a two-vehicle platoon. Note that the distance between the two platooning vehicles is much smaller than the distance between the non-platooning vehicles.

To assess the quality of the evolved neural net controllers, we also performed benchmark experiments. In these experiments, the vehicles are controlled using only the IDM behavioural model. We did 100 simulation runs with these settings. In Table 5.1, all experimental settings are summarised. Since we present this work mainly as a proof-of-concept, we did not perform any thorough parameter analysis of NEAT.

In Figures 5.4 and 5.5, two example screen shots of the simulation are given. Note that these are very zoomed-in screen shots and therefore only show a very small part of the total simulation.
5.1. Evolving Intelligent Vehicle Control using Multi-Objective NEAT

Figure 5.5 – A join scenario. The light-blue vehicle was already part of the platoon and the dark blue vehicle wants to join. After the light-blue vehicle has made sufficient space, the blue vehicle will change lanes and be part of the platoon.

Figure 5.6 – The inputs and outputs of the neural network vehicle controllers. The inputs are observables from the simulation, the output is the action that has to be performed. The neural network in the middle is evolved by the NEAT algorithm.
In Figure 5.7, we show the resulting Pareto front that we obtained from all runs. On the x-axis, the travel time objective is shown and on the y-axis the comfort objective, measured in mean number of lane changes.

In this figure, we can see the Pareto-optimal solutions, depicted by the white squares. This indicates that indeed, some controllers perform well in the travel time objective and less in the comfort objective and vice versa.

After we found these Pareto-optimal solutions, we took those controllers and tested them again in order to find how stable these controllers are. The results of these experiments can be seen in the same chart, they are the items indicated by a black plus. These results show that the reruns unfortunately are not as stable as we hoped.

However, when we compare the results with some experimental runs in which the vehicles only use the IDM controllers, it shows that all evolved controllers perform better on both objectives then the IDM controller, which is a very promising result.

We have omitted one Pareto-optimal solution from this chart, which had a good travel time objective value, but a very bad comfort objective value. Furthermore, when we retested this controller, its results all fell far outside the range of the Pareto-optimal solutions.

We learn from these results that there is much work to be done in terms of creating solutions that perform stable in the Pareto-optimal regions of the fitness landscape. This could be done by running multiple fitness evaluations per controller. This way, we eliminate the controllers that are overall worse, but had a single lucky run during our experiments. Furthermore, in our current experiments, there might be many evolved controllers that are slightly worse than Pareto-optimal. This means that these controllers did not show up in our results. But, it might be that these controllers yield a much more stable performance. By running more fitness evaluations per controller, we can find these controllers more easily. Moreover, we could include controller stability as an explicit fitness measure, making it even easier to find the more stable controllers.

As a proof-of-concept, we show with these results that indeed it is possible to yield a whole range of vehicle controllers that each reflect their prioritisation of preference measures. The method can easily be extended with extra fitness measures, such as fuel economy or additional comfort measures, such as the jerk.
of the vehicle. We primarily want to show that this method works in principle, which turns out to be the case.

The resulting controllers perform better than the IDM controller, which is depicted in the chart by the cross in the top-right corner. This means that automatic driving is an excellent alternative to manual driving.

### 5.1.6 Conclusions and Future Work

In this section, we envision a future in which vehicles on highways are able to drive autonomously, using Cooperative Adaptive Cruise Control. With this CACC technology, vehicles are able to form trains that drive at very short headway times, also known as platoons. In this study, we did preliminary experiments to determine if platooning strategies can be evolved, with the goal that the drivers of the vehicles can adjust these evolved driving strategies of the vehicle, based on their personal preferences.
We have showed promising preliminary results. Our multi-objective version of the NEAT algorithm is able to find a set of controllers that each perform in a different area on the Pareto front. The controllers that we found also outperform the benchmark that we used, which is a controller that uses the IDM longitudinal model and the MOBIL lane-changing model. This means that in terms of our defined objectives, our method works very well.

Since this study primarily serves as a proof-of-concept, we can conclude that the results that we obtained are very promising. It is a generic method that can easily be extended with extra objectives, additional input variables or more actions for the vehicles.

The work that we describe in this section is a starting point for further investigations in these techniques. First, we will try to find more stable controllers. Currently, the resulting controllers have great variance in performance, which might be due to the number of runs that we performed. Also, we plan to do experiments with even more objectives, such as fuel economy and the stability of the controller.

Another track of future work includes doing tests with heterogeneous vehicles. In our current work, we tested a particular neural network by letting all vehicles in a simulation use the same network. The question remains whether these networks remain their performance when other vehicles have completely different controllers on board.

Our preliminary results are very promising, but our conclusions need to be confirmed by more extensive experimentation, both in terms of complexity of the experiments and the sheer number of experiments that we want to run. Work to this end is described in the next sections.
5.2 A Multi-Objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems

The research in this section is inspired by a vision of intelligent vehicles that autonomously move along motorways: they join and leave trains of vehicles (platoons), overtake other vehicles, etc. We propose a multi-objective evolutionary algorithm based on NEAT and SPEA2 that evolves high-level controllers for such intelligent vehicles. The algorithm yields a set of solutions that each embody their own prioritisation of various user requirements such as speed, comfort or fuel economy. This contrasts with the current practice in researching such controllers, where user preferences are summarised in a single number that the controller development process then optimises.

Proof-of-concept experiments show that evolved controllers substantially outperform a widely used human behavioural model. We show that it is possible to evolve a set of vehicle controllers that correspond with different prioritisations of user preferences, giving the driver, on the road, the power to decide which preferences to emphasise.

5.2.1 Introduction

In Europe’s major cities, the average driver loses as much as 60 to 70 hours per year in traffic jams,\textsuperscript{iii} the cause of tremendous environmental and socio-economical impact. Vehicle manufacturers are introducing Adaptive Cruise Control (ACC) to help combat this issue. ACC allows the driver to set the desired distance to the


Section 5.2 was published as:

preceding vehicle. The ACC system measures the vehicle’s relative distance and velocity through radar and adjusts speed accordingly. Such automated maintenance of distance is hoped to increase the uniformity of traffic flows and so better utilise road capacity.

Cooperative Adaptive Cruise Control (CACC) is a major development in recent research on Intelligent Transportation Systems (ITS) that takes this automated control to the next level by adding direct communication between vehicles (see Figure 5.8). This enables the vehicles to directly communicate about their current velocity, position and acceleration and so removes some of the uncertainty and delays that relying only on radar measurements implies. Directly communicating such accurate state information allows vehicles to drive much closer to each other without compromising safety. This opens up the possibility of platooning: trains of vehicles that drive very close to each other at (near) equal speed and therein lies one of the big selling points of CACC. It enables platoons to be string stable: changes in acceleration or deceleration are reduced by the following vehicles instead of amplified. This property is expected to greatly improve the throughput of vehicles on highways, because it is exactly the amplification of acceleration and deceleration that causes many traffic jams\textsuperscript{iv}.

\textsuperscript{iv}See \url{http://www.drivenbyhelmond.nl/en/projecten/tno-doet-onderzoek-naar-spookfiles/}
CACC will become available in the near future. Vehicles will be equipped with a simple button that turns the CACC mode on and off again. With the system turned on, the vehicle will then, for instance, search for the nearest platoon available to join and (almost) automatically drive along in the platoon until the driver decides to take matters into her own hands and turn CACC off again. Such CACC systems do not yet take into account the preferences of specific drivers. Some drivers like to drive fast, while others prefer fuel economy and care less about speed. Ride comfort is also a consideration that can be measured in number of lane changes, changes in velocity, etc.

We envision a system where the driver can enter her preferences into the system at real time. Setting preferences corresponds with selecting a high-level vehicle controller. These controllers choose actions (e.g., joining, leaving or creating platoons, changing lanes or simply driving on) based on a number of input variables such as acceleration, velocity and distance to preceding vehicle. This vision is at odds with most current research into controller development because of the way driver preferences are typically taken into account: the preferences are all summarised into one single metric (e.g. a weighted average) that is then optimised.

We propose to use multi-objective evolutionary algorithms as an approach to develop controllers without having to summarise the separate preference metrics into a single number. This approach yields a number of Pareto-optimal controllers that each reflect their own prioritisation of preference measures. Some may be fast but less comfortable, others may be very comfortable but slow, yet others may strike another balance between the user preferences. Such a selection of controllers would enable a user to set the relative importance of fuel economy, speed, lane changes, etc. The controller that best approximates the current prioritisation of preferences is then activated and vehicle progresses as the user has requested. Thus, we seek the benefit of evolutionary multi-objective optimisation of autonomous vehicle controllers not so much in better control (although that would of course not be disregarded), but more in the fact that it would yield multiple controllers with varying emphasis on the different user preferences and so allow the user to select and change the appropriate vehicle behaviour while driving. The method we propose employs NEAT, a well-known algorithm for neuro-evolution (Stanley and Miikkulainen, 2002) and combines it with the Pareto Strength approach as found in the SPEA2 evolutionary algorithm (Zitzler et al., 2002). It is
in itself not a new multi-objective evolutionary algorithm, but an easy way for multi-objectivation of an otherwise single-objective evolutionary algorithm.

As a proof-of-concept, we conduct a number of simulated experiments in which we evolve controllers to optimise both speed and comfort as twin objectives.

Our first research question is whether our NEAT-Pareto Strength (NEAT-PS) approach does indeed yield a useful set of controllers that embody different prioritisations of these objectives. Secondly, we hypothesise that the evolved controllers, by virtue of their platooning behaviour, yield better speed as well as comfort compared to a benchmark controller based on the Intelligent Driver Model (IDM) (Treiber et al., 2000) in combination with the MOBIL lane changing model (Treiber and Helbing, 2002). These two models are commonly used to model human drivers in traffic flow modelling.

In previous work (van Willigen et al., 2013a), we presented preliminary results that showed the potential of this technology. In this section, we present a more thorough investigation of the technique. By modifying the inputs and outputs of the neural network controller of the vehicles and by increasing the number of fitness evaluations per individual, we managed to get a much better performance of the algorithm. The resulting Pareto front clearly distinguishes between the two objectives. Furthermore, we validated the results by re-running the Pareto-optimal solutions and these validation reruns lie within range of the original Pareto-optimal solutions, indicating that these Pareto-optimal controllers perform consistently.

5.2.2 Related Work

While ACC systems are currently being adopted in consumer vehicles, research and development into cruise control focuses on enabling more and better cooperation between ACC systems, yielding CACC systems. Van Arem et al. (2006) describe the effect of CACC on traffic flow. They conclude that, when the penetration level of CACC-equipped vehicles is high enough (> 60%), traffic stability and throughput is improved. In Yang et al. (2004), a communication protocol is proposed in order to make a cooperative collision warning system on highways.

The main suggested application area of CACC technology these days is platooning. Broggi et al. (2000) and Kanellakopoulos et al. (1999) both use image recognition techniques in combination with sensors to autonomously enable pla-
A Multi-Objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems

However, current technology has improved significantly since then and nowadays direct radio communication between vehicles is used to enable platooning.

Naus et al. (2010) thoroughly investigate the issue of string stability in platooning, with both ACC and CACC controllers. Their method includes several factors of delay in communication.

In Hallé (2005), an extensive architecture is given for a layered multi-agent CACC architecture. The authors use this architecture to implement both centralised platoons (in which there is a coordinating platoon leader) and decentralised platoons (in which all vehicles operate as equals). Khan et al. (2008) present different platoon (in their paper, convoy) forming strategies, based on a utility value of a platoon.

There has not been much work yet on the development of platooning strategies, where the vehicles have to decide while driving if and when they should be platooning. In the SARTRE project (Safe Road Trains for the Environment) (Bergenhem et al., 2010), platoons are defined as trains of road vehicles of which the front one is a trained platoon driver. This project has implemented basic platooning strategies, also in real vehicles, but these strategies are only defined at the action level, where for example the join and leave actions are defined, but not at a higher level, where a strategy for driving on the highway is defined.

The multi-objective driving strategies domain has been investigated by Dovgan et al. (2011, 2012). They use the NSGA-II multi-objective evolutionary algorithm to determine driving strategies that optimise travel time and fuel economy. Other vehicles are not taken into account in this work, only the vehicle’s own state space and route state space. Platooning is not an application that is taken into account.

In the domain of human driver modelling, several well-known models exist, such as the simple yet elegant Intelligent Driver Model (Treiber et al., 2000) and the older Gipps model (Gipps, 1981).

Neuro-evolution is a form of machine learning that uses evolutionary algorithms to train artificial neural networks. There are many methods for neuroevolution, with NEAT (Stanley and Miikkulainen, 2002) and its extension HyperNEAT (Gauci and Stanley, 2007) as fairly recent notable developments.

There are also many multi-objective evolutionary algorithms; two well-known state-of-the-art approaches are SPEA2 (Zitzler et al., 2002) and NSGA-II (Deb et al., 2002). SPEA2 calculates a single scalar fitness value based on the number of other
individuals an individual dominates—the Pareto strength—which is then used for parent and survivor selection. NSGA-II uses a Pareto ranking mechanism, also based on the dominance relation, but doesn’t attach specific fitness values, making it harder to reconcile with NEAT’s standard selection and niching mechanisms.

5.2.3 Algorithm

NEAT (Stanley and Miikkulainen, 2002) is a popular state-of-the-art algorithm for evolving neural networks which also optimises the network topologies. Thus, we conveniently do not have to decide on a topology for the neural network beforehand. This also allows for the algorithm to propose functionally different neural networks. To achieve this, it employs a clever method of speciation of the neural network topologies, taking into account specific sub-structures within the topology. The neural networks compete primarily with individuals within their own niches instead of with individuals of the entire population. This technique makes it easier to protect innovations within the topologies.

To achieve our vision, it is essential to take multiple objectives into account. The original NEAT algorithm, however, does not support multiple objectives. Therefore, we extend the NEAT algorithm to allow for multi-objective optimisation. We incorporated the Pareto Strength approach from SPEA2 into NEAT to create NEAT-PS. The advantage of the Pareto Strength is that it computes a single fitness value for each individual, based on the values of all objective functions, making it very straightforward to augment the NEAT algorithm with multi-objective capabilities.

Our proof of concept implementation uses two objectives: travel time and comfort. These objectives are somewhat incompatible, so controllers must prioritise the objectives in varying ways: some are fast but less comfortable, others are slower but much more comfortable. Controllers with varying prioritisations would allow the driver to balance the preferences as she wants at any time. Some users prefer speed, others prefer comfort and this varies with their circumstances. By allowing the driver to switch between alternative controllers (by prioritising objectives) while driving, we allow for changes in driving behaviour to match the current preferences of the person behind the wheel.

The basis of the Pareto Strength evaluation is the domination relation:
In this formula, the $\succeq$ relation between objectives only specifies that objective $o_i$ is equal or better than objective $o_j$, without saying whether the objective is maximised or minimised. In our specific experiments, we are minimising both objectives.

After calculating each objective function’s value for all individuals, we calculate the Pareto Strength values for each individual $i$ in population $P$ as described in Zitzler et al. (2002):

$$S(i) = |\{j | j \in P \land i \succ j\}|$$

(5.6)

where $|\cdot|$ is the cardinality of a set and $\succ$ is the dominance relation. Now we can calculate the raw fitness values of each individual $i$ by summing the strengths of the individuals $j$ that dominate $i$:

$$R(i) = \sum_{j \in P, j \succ i} S(j)$$

(5.7)

The final fitness value for an individual that we feed into the regular NEAT algorithm is then given by:

$$F(i) = \frac{10000}{1 + R(i)}$$

(5.8)

From these formulae, it follows that if an individual is not dominated, its raw fitness value is 0 and the final fitness value has the maximum value of 10,000. This evaluation creates a single fitness value that reflects multiple fitness metrics and we can straightforwardly plug these results into the standard NEAT algorithm\textsuperscript{v} to complete NEAT-PS.

### 5.2.4 Experimental Setup

Our experimental setup consists of two main parts: the vehicle simulator in which we tested our controllers and NEAT-PS which evolves the vehicle controllers based on the feedback it obtains from the simulator. This setup is schematically depicted in Figure 5.9.

\textsuperscript{v}We used the NEAT4j implementation, http://neat4j.sourceforge.net
Figure 5.9 – Graphical representation of our simulation loop. The NEAT algorithm generates a new population of controllers. Each member of a new generation of vehicle controllers is then evaluated by the vehicle simulation. The simulator assigns the fitness values (the Pareto strengths) to the controllers for NEAT to use in its parent and survivor operators. The loop is repeated for a preset number of generations.

We used Movsim (Kesting, 2008), an open-source vehicle simulator that implements the Intelligent Driver Model (IDM) and MOBIL, which are respectively the longitudinal (determining minimum headway and consequent speed adjustments) model and lane changing model that form a benchmark in our experiments. Movsim is primarily a microscopic simulator of traffic, whereas we want to use higher-level actions as input for the vehicle (such as creating, joining and leaving a platoon). Therefore we had to extend the functionality of the simulator to provide these higher-level actions. In the original simulator, each vehicle can only use a single longitudinal controller and this controller decides on every timestep what the acceleration of the vehicle should be.

We extended the simulator so that each vehicle can use many controllers; a decision making process (such as a neural network) on a more abstract level selects the next action of the vehicle based on the current state of the vehicle. To execute the selected action the vehicle decides on the appropriate longitudinal controller for that moment. For example, when joining a platoon, the vehicle first has to drive towards the platoon using a specific longitudinal controller that facilitates this. Then, when the vehicle has approached the platoon, it may have to change to the correct lane. Other vehicles may have to make space for the joining vehicle. All these processes correspond with different longitudinal controllers. Figure 5.10 schematically shows the extra functionality of the simulator. The original Movsim
A Multi-Objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems

Figure 5.10 – The five-layered architecture of our vehicle controllers. In the original simulation settings, the three layers in the middle are missing, thus calculating the acceleration directly based on the input variables.

Implementation directly links inputs to acceleration; we introduce three intermediate layers to facilitate behaviours on a more abstract level.

The higher level actions implement realistic platooning behaviour, where vehicles on the highway drive very close to the preceding vehicles while copying the preceding vehicle’s speed and acceleration. The intricacies of the platooning control, such as sensing, information fusing and communication models are not taken into account in this simulation: these issues should be solved at the lower levels of the system. They are not within the scope of this work and we assume these technologies to be working correctly and without error.

Figures 5.11 and 5.12 show two example screen shots of the simulation. Note that these are very zoomed-in screen shots and therefore only show a very small part of the total simulation.

5.2.4.1 Controller

By default, the vehicles in our simulation drive according to the Intelligent Driver Model (IDM) and the MOBIL lane changing model. Additionally, our vehicles can execute actions, each triggering other versions of longitudinal controllers.

Create Platoon If the vehicle is currently not part of a platoon, it creates a new platoon. The IDM longitudinal model remains active and the lane changing model is turned off (in our simulation, platoons cannot change lanes). The
vehicle that creates the platoon automatically becomes the platoon leader – the foremost vehicle in the platoon.

**Join Platoon Front** If the vehicle is currently not part of a platoon, it tries to join a nearby platoon. This action checks if there is a platoon directly in front of the vehicle. If so, the vehicle drives up to the last vehicle in that platoon and changes its longitudinal model to CACC, which copies the velocity and acceleration of the preceding vehicle.

**Join Platoon Side** This action joins a platoon that is directly next to the vehicle. First, the vehicle checks if there is a platoon on its side and if so, the platoon makes way for the vehicle to join. As soon as there’s enough space for this, the vehicle switches lanes and joins the platoon by switching to the CACC longitudinal model.

**Figure 5.12** – A join scenario. The light-blue vehicle was already part of the platoon and the dark blue vehicle wants to join. After the light-blue vehicle has made sufficient space, the blue vehicle will change lanes and be part of the platoon.
5.2. A Multi-Objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems

**Leave Platoon** If the vehicle is part of a platoon, the vehicle leaves the platoon and starts driving using the IDM and MOBIL models again. Immediately, more headway time is created. Additionally, if the vehicle that leaves the platoon was driving somewhere in the middle of the platoon, the platoon is split into two parts, being the sub-platoon in front of the vehicle and the sub-platoon behind the vehicle. If the leaving vehicle was platoon leader, this role is transferred to the second vehicle in the platoon. If the leaving vehicle was the only vehicle in the platoon, the platoon is disbanded. When a vehicle leaves the highway, it also leaves the platoon.

**Change Lane** If possible, the vehicle changes lanes. Our simulation currently consists of two lanes, so this action does not need a parameter telling which lane to change to.

**Do Nothing** The vehicle longitudinal controller remains unchanged.

In our current scenario, the vehicles drive completely autonomously and according to the decisions of the neural network. However, human drivers should be able to override any given action that the neural network comes up with or to turn off the system altogether, much like current implementations of (adaptive) cruise control.

### 5.2.4.2 Neural Network

NEAT-PS evolves the neural network that selects from the six actions listed above. The network inputs and outputs are depicted in Figure 5.13, listing the following input variables for a vehicle $c$:

- Headway time of vehicle to preceding vehicle: The distance to the preceding vehicle divided by the velocity of $c$;

- Fraction of front platoon speed and desired velocity: The speed of the platoon in front of $c$ (if there is one) divided by $c$’s desired velocity;

- Fraction of side platoon speed and desired velocity: The speed of the platoon on the side of $c$ (if there is one) divided by $c$’s desired velocity;

- Fraction of desired velocity: The current velocity of $c$ divided by its desired velocity;
Chapter 5. Preference-based Controller Building

- isPlatooning?: The current platooning status of c;

- Headway time to preceding vehicle on other lane: the distance to the first preceding vehicle on the other lane divided by c’s velocity.

The input variables can easily be modified or extended with other variables, but it has to be kept in mind that the vehicle should be able to register these variables autonomously and on a real-time basis. The values of the input variables are normalised to \([0, 1]\) (except those inputs that are a fraction of the desired speed, those can be greater than 1) to make them more amenable as inputs for the neural network.

The six output variables select from the actions that we have defined in our simulation: the controller simply picks the action with the highest corresponding output value.

**Figure 5.13** – The input and output layers of the neural network controller of the vehicles. The inputs are observables from the simulation and the outputs are the actions that need to be performed (if possible). The neural network layers in between the input and output layers are evolved by NEAT.

### 5.2.4.3 Evaluation Criteria

The experiments described in this section serve primarily as a proof-of-concept and we limit ourselves to using two objectives:
Comfort As a measure of the comfort of a ride, we take the number of times a vehicle changes lanes during a ride, assuming that fewer lane changes indicates a more comfortable ride.

Speed Our second objective in this study is speed, simply measured as the time it takes a vehicle to complete the stretch of motorway in our simulation.

To evaluate each controller, we simulate a 10 km, two-lane stretch of road with 3000 vehicles entering per hour, all running that particular neural network. Each vehicle has a desired speed of 30m/s. Their initial speeds differ by a random amount up to 5m/s from the desired speed.

Each vehicle evaluates the neural network every 5 seconds and executes the action that the neural network proposes. The vehicle only executes the action if it is possible. For example, the vehicle will not change lanes if this would result in a crash. If an action is not executable, the vehicle will simply remain using the longitudinal controller it was already using, waiting another 5 seconds before re-evaluating the neural network.

We run this simulation for 10 minutes of simulation time (i.e. ca. 500 vehicles enter the stretch of road), after which we measure the mean number of lane changes of the vehicles and the mean travel times of the vehicles that completed the 10km stretch. These measurements are then used to calculate the controller’s Pareto Strength as described above.

In this setting, we ran 50 repeats of 50 generations each with population size 50. From the 50 final generations that obtained from these runs, we extract the combined Pareto-optimal solutions.

To assess the quality of the evolved neural net controllers, we also performed 100 repeats of a benchmark experiment where the vehicles are controlled using only the IDM behavioural model.

Table 5.2 summarises the experimental settings.

5.2.5 Results and Analysis

Figures 5.14 and 5.15 show the Pareto-optimal solutions resulting from the 50 repeats of the NEAT-PS experiment. This selection consists of the non-dominated controllers from the union of the non-dominated controllers per run. The ‘×’ ticks indicate the performance in travel time and comfort for each of these controllers,
Table 5.2 – Experimental Setup

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road length</td>
<td>10km</td>
</tr>
<tr>
<td>Vehicle inflow</td>
<td>3000veh/hr</td>
</tr>
<tr>
<td>Desired speed</td>
<td>approx. 30m/s</td>
</tr>
<tr>
<td>Max deceleration</td>
<td>-3m/s²</td>
</tr>
<tr>
<td>Simulation time</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Timestep size</td>
<td>0.20 seconds</td>
</tr>
<tr>
<td>Neural Network Evaluations</td>
<td>Every 5 seconds</td>
</tr>
<tr>
<td>Neural Network inputs</td>
<td>6</td>
</tr>
<tr>
<td>Neural Network outputs</td>
<td>6</td>
</tr>
<tr>
<td>Population Size</td>
<td>50</td>
</tr>
<tr>
<td>Fitness Evaluations per individual</td>
<td>10</td>
</tr>
<tr>
<td>Generations</td>
<td>50</td>
</tr>
<tr>
<td>Repeats</td>
<td>50</td>
</tr>
<tr>
<td>Benchmark runs</td>
<td>100</td>
</tr>
<tr>
<td>Validation runs</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 5.14 – All experimental results. The original Pareto front is show by ‘×’ signs. The crossed error bars show the medians and quartile performance of the validation reruns. The crossed error bars in the top right show the IDM results. This controller clearly performs much worse than the evolved controllers.

averaged over 10 runs as explained in Section 5.2.4.3. In these plots, the x-axis shows the travel time in seconds, the y-axis the level of comfort measured as the mean number of lane changes over the 10km stretch.

The controllers are evenly spread along the Pareto front; if these controllers were to be implemented (putting aside the issue of any reality gap for the moment), the user would indeed be offered a range of choices to strike a balance between speed and comfort.
5.2. A Multi-Objective Approach to Evolving Platooning Strategies in Intelligent Transportation Systems

Figure 5.15 – The experimental results, excluding the IDM results, for clarity reasons.

Figure 5.14 also shows median and quartile performance on both objectives for 100 trials with different random seeds with the benchmark IDM controller. Because this controller performs at a different level from the solved controllers, Figure 5.15 shows the results for the NEAT-PS solutions in more detail.

When we compare the results of the evolved controllers with those of the benchmark IDM controller, it is clear that all the Pareto-optimal controllers perform substantially better on both objectives.

To validate the stability of performance of the solutions on the final combined Pareto front, we subsequently ran a further 100 trials with different random seeds for each of these solutions. The crossed error bars in Figures 5.14 and 5.15 show the median and upper and lower quartile performance from these validation runs. While these rerun results lie close to the original Pareto front, they are, in all cases worse than the performance measured during evolution.

This difference is highlighted in Figure 5.16, which shows the results from the 100 reruns for a randomly selected Pareto-optimal controller. The crossed error bars are close to the highlighted average performance during evolution. We think that the reason for this skewed expectation of performance is due to the way in which we combine the evaluation results during evolution – by taking the mean value of ten trials. A more sophisticated comparison of performance distribution, for instance by redefining the domination relationship as ‘being significantly better’ (with some $\alpha$), might alleviate this problem. This approach is advocated in Smit and Eiben (2011) for comparing multi-objective evolutionary algorithm runs. Alternatively, more consistent performance may be achieved by adding consistency as an additional objective. It must be noted that this consistency objective should not be an explicit objective for the user, but rather an implicit objective for
the algorithm itself.

5.2.6 Conclusions and Future Work

We described our vision of a future in which vehicles on highways can drive autonomously using Cooperative Adaptive Cruise Control. With CACC technology, vehicles are able to form trains of vehicles that drive very close to one another, also known as platoons. We performed proof-of-concept experiments that show that multi-objective evolutionary algorithms can evolve a number of controllers that embody a variety of prioritisations of ride qualities like speed and comfort. Such a combination of controllers will, in our vision, allow the driver to emphasise different qualities based on her preference and needs in real time.

The experiments described in this study serve primarily as a proof-of-concept and therefore we limited ourselves two objectives: speed and comfort. These two objectives are easily extended with more objectives, such as fuel economy (when a vehicle is platooning, it consumes less fuel than otherwise) or further measures for comfort (e.g. the variability of speed of the vehicle). We plan to consider such additional objectives in future work.

The evolved controllers outperform the benchmark IDM controller substantially. The IDM is a widely used model of human driver behaviour; we conclude that automatic driving as we envisage will substantially increase travel speed as well as ride comfort.

These preliminary results are very promising, but our conclusions need to be confirmed by more extensive experimentation, both in terms of complexity and number of experiments. Work to this end is underway.

![Figure 5.16](image.png)

**Figure 5.16** – Original performance with raw and summarised validation results.
The concept of evolving many Pareto-optimal controllers represents a significant improvement from the current practice of optimising a weighted average of objectives to obtain a single controller that prioritises ride qualities as the developer deems appropriate. The consequent freedom of choice for drivers to select controller behaviour as they see fit will greatly benefit acceptance of CACC technology.

These results open up possibilities for substantial further research. Firstly, we plan to do experiments with more objectives, such as fuel economy and other ways of measuring comfort in a vehicle. Secondly, we will include stability of the controller as an explicit objective in the NEAT algorithm.

Another track of future work includes tests with heterogeneous vehicles. In our current work, we tested a particular neural network by letting all vehicles in a simulation use the same network. The question remains whether these networks remain their performance when other vehicles have completely different controllers or even human drivers on board.
5.3 Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives

The research in this section is inspired by a vision of intelligent vehicles that autonomously move along motorways: they join and leave trains of vehicles (platoons), overtake other vehicles, etc. We propose a multi-objective evolutionary algorithm that evolves high-level controllers for such intelligent vehicles. The algorithm yields a set of solutions that each embody their own prioritisation of various user requirements such as speed, comfort or fuel economy. This contrasts with the current practice in researching such controllers, where user preferences are summarised in a single number that the controller development process then optimises.

In this section, we test our multi-objective approach on 6 objectives. Our method outperforms a widely used human behavioural model on many of the objectives. Some performance is lost when we introduce additional objectives, but these losses are small and therefore acceptable. We show that it is possible to evolve a set of vehicle controllers that correspond with different prioritisations of user preferences, giving the driver, on the road, the power to decide which preferences to emphasise, although we do see that the more objectives are added to the system, the less intuitive the prioritisation of the different objectives becomes.

5.3.1 Introduction

In Europe’s major cities, the average driver loses as much as 60 to 70 hours per year in traffic jams,\textsuperscript{vi} the cause of tremendous environmental and socio-economical impact. Vehicle manufacturers are introducing Adaptive Cruise Control (ACC)


Section 5.3 was published as:

acc allows the driver to set the desired distance to the preceding vehicle. the acc system measures the vehicle’s relative distance and velocity through radar and adjusts its speed accordingly. such automated maintenance of distance is hoped to increase the uniformity of traffic flows and so better utilise road capacity.

cooperative adaptive cruise control (cacc) is a major development in recent research on intelligent transportation systems (its) that takes this automated control to the next level by adding direct communication between vehicles (see figure 5.17). this enables the vehicles to directly communicate about their current velocity, position and acceleration and so removes some of the uncertainty and delays that relying only on radar measurements implies. directly communicating such accurate state information allows vehicles to drive much closer to each other without compromising safety. this opens up the possibility of platooning: trains of vehicles that drive very close to each other at (near) equal speed, and therein lies one of the big selling points of cacc: the capacity of highways is increased, reducing traffic jams and increasing driver safety and comfort. furthermore, this method of reducing traffic jams is extremely cheap compared to the current trend of increasing highway capacity by adding extra lanes. also, the fact that this technology can be developed without interference of government makes it very attractive.

(a) acc scenario - vehicles have an on-board radar that detects information about the preceding vehicle.

(b) cacc scenario - vehicles also have a radio unit on-board, to communicate directly with nearby vehicles (not only with the preceding vehicle).

figure 5.17 – a schematic representation of the difference between acc and cacc.
CACC will become available in the near future. Vehicles will be equipped with a simple button that turns the CACC mode on and off again. With the system turned on, the vehicle will then, for instance, search for the nearest platoon available to join and (almost) automatically drive along in the platoon until the driver decides to take matters into her own hands and turn CACC off again. Such CACC systems do not yet take into account the preferences of specific drivers.

We envision a system where the driver can enter her preferences into the system at real time. Setting preferences corresponds with selecting a high-level vehicle controller. These controllers choose actions (e.g., joining, leaving or creating platoons, changing lanes or simply driving on) based on a number of input variables such as acceleration, velocity, and distance to preceding vehicle. This vision is at odds with most current research into controller development because of the way driver preferences are typically taken into account: the preferences are all summarised into one single metric (e.g. a weighted average) that is then optimised.

We propose to use multi-objective evolutionary algorithms as an approach to develop controllers without having to summarise the separate preference metrics into a single number. This approach yields a number of non-dominated controllers that each reflect their own prioritisation of preference measures. For example, some may be fast but less comfortable, others may be very comfortable but slow, yet others may strike another balance between the user preferences. Such a selection of controllers would enable a user to set the relative importance of fuel economy, speed, lane changes, etc. The controller that best approximates the current prioritisation of preferences is then activated and vehicle progresses as the user has requested. Thus, we seek the benefit of evolutionary multi-objective optimisation of autonomous vehicle controllers not so much in better control (although that would of course not be disregarded), but more in the fact that it would yield multiple controllers with varying emphasis on the different user preferences and so allow the user to select and change the appropriate vehicle behaviour while driving. The method we propose employs NEAT, a well-known algorithm for neuro-evolution (Stanley and Miikkulainen, 2002) and combines it with the Pareto Strength approach as found in the SPEA2 evolutionary algorithm (Zitzler et al., 2002). We conduct a number of simulated experiments in which we evolve controllers to optimise 6 different objectives simultaneously.

In earlier work (van Willigen et al., 2013c), we showed that the NEAT-Pareto Strength (NEAT-PS) approach did indeed yield a useful set of controllers that
embodied prioritisation of 2 different objectives. In this section, we extend the method with 4 additional objectives. These additional objectives include fuel economy, an extra comfort measure, the number of failed actions and the consistency of a controller. These extra objectives are explained in detail in section 5.3.4.

Our first research question is whether or not our earlier tested NEAT-PS approach still holds when drastically increasing the number of objectives that have to be optimised. We compare our results to the version in which we optimised only two objectives. Secondly, we want to see how different objectives interact with each other. We hypothesise that some objectives are heavily correlated, while others are not correlated at all. Thirdly, we hypothesise that the evolved controllers outperform a benchmark controller based on the Intelligent Driver Model (IDM) (Treiber et al., 2000) in combination with the MOBIL lane changing model (Treiber and Helbing, 2002), not only on the speed and number of lane changes, but also on the other applicable objective measures that we used in this study.

5.3.2 Related Work

While ACC systems are currently being adopted in consumer vehicles, research and development into cruise control focuses on enabling more and better cooperation between ACC systems, yielding CACC systems. Van Arem et al. (2006) describe the effect of CACC on traffic flow. They conclude that, when the penetration level of CACC-equipped vehicles is high enough (> 60%), traffic stability and throughput is improved. In Yang et al. (2004), a communication protocol is proposed in order to make a cooperative collision warning system on highways.

The main suggested application area of CACC technology these days is platooning. Broggi et al. (2000) and Kanellakopoulos et al. (1999) both use image recognition techniques in combination with sensors to autonomously enable platooning. However, current technology has improved significantly since then, and nowadays direct radio communication between vehicles is used to enable platooning.

In Hallé (2005), an extensive architecture is given for a layered multi-agent CACC architecture. The authors use this architecture to implement both centralised platoons (in which there is a coordinating platoon leader) and decentralised platoons (in which all vehicles operate as equals). Khan et al. (2008)
present different platoon (in their paper, convoy) forming strategies, based on a utility value of a platoon.

There has not been much work yet on the development of platooning strategies, where the vehicles have to decide while driving if and when they should be platooning. In the SARTRE project (Safe Road Trains for the Environment) (Bergenhem et al., 2010), platoons are defined as trains of road vehicles of which the front one is a trained platoon driver. This project has implemented basic platooning strategies, also in real vehicles, but these strategies are only defined at the action level, where for example the join and leave actions are defined, but not at a higher level, where a strategy for driving on the highway is defined.

The multi-objective driving strategies domain has been investigated by Dovgan et al. (2011, 2012). They use the NSGA-II multi-objective evolutionary algorithm to determine driving strategies that optimise travel time and fuel economy. Other vehicles are not taken into account in this work, only the vehicle’s own state space and route state space. Platooning is not an application that is taken into account.

In the domain of human driver modelling, several well-known models exist, such as the simple yet elegant Intelligent Driver Model (Treiber et al., 2000) and the older Gipps model (Gipps, 1981).

Neuro-evolution is a form of machine learning that uses evolutionary algorithms to train artificial neural networks. There are many methods for neuro-evolution, with NEAT (Stanley and Miikkulainen, 2002) and its extension Hyper-NEAT (Gauci and Stanley, 2007) as fairly recent notable developments. Hyper-NEAT uses an extra layer of encoding to represent higher-level pattern structures such as symmetry and repetition. HyperNEAT thus utilises geometric regularities of the task domain, making it less useful with our neural networks, that do not need to represent any geometric regularities.

There are also many multi-objective evolutionary algorithms; two well-known state-of-the-art approaches are SPEA2 (Zitzler et al., 2002) and NSGA-II (Deb et al., 2002). SPEA2 calculates a single scalar fitness value based on the number of other individuals an individual dominates –the Pareto strength– which is then used for parent and survivor selection. NSGA-II uses a Pareto ranking mechanism, also based on the dominance relation, but doesn’t attach specific fitness values, making it harder to reconcile with NEAT’s standard selection and niching mechanisms.
5.3. Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives

5.3.3 Algorithm

NEAT (Stanley and Miikkulainen, 2002) is a popular state-of-the-art algorithm for evolving neural networks which also optimises the network topologies. Thus, we conveniently do not have to decide on a topology for the neural network beforehand. This also allows for the algorithm to propose functionally different neural networks. To achieve this, it employs a clever method of speciation of the neural network topologies, taking into account specific sub-structures within the topology. The neural networks compete primarily with individuals within their own niches instead of with individuals of the entire population. This technique makes it easier to protect innovations within the topologies.

To achieve our vision of optimising multiple user preferences, it is essential to take multiple objectives into account. The original NEAT algorithm, however, does not support multiple objectives. Therefore, we extend the NEAT algorithm to allow for multi-objective optimisation. We incorporated the Pareto Strength approach from SPEA2 into NEAT to create NEAT-PS. The advantage of the Pareto Strength is that it computes a single fitness value for each individual, based on the values of all objective functions, making it very straightforward to augment the NEAT algorithm with multi-objective capabilities.

Our implementation uses six objectives: travel time, two measures for comfort, fuel economy, minimising the number of failed actions in the controller and the consistency of the controllers. These objectives may or may not be compatible, so controllers must prioritise the objectives in varying ways: for example, some are fast but less comfortable, others are slower but much more comfortable. Controllers with varying prioritisations would allow the driver to balance the preferences at any time. Some users prefer speed, others prefer comfort or fuel economy, and this varies with their circumstances. By allowing the driver to switch between alternative controllers (by prioritising objectives) while driving, we allow for changes in driving behaviour to match the current preferences of the person behind the wheel.

The basis of the Pareto Strength evaluation is the domination relation:

\[ i \succ j \iff \forall o \in O : o_i \succeq o_j \land i \neq j \]  \hspace{1cm} (5.9)
In this formula, the $\succeq$ relation between objectives only specifies that objective $o_i$ is equal or better than objective $o_j$, without saying whether the objective is maximised or minimised. In our specific experiments, we are minimising all objectives.

After calculating each objective function’s value for all individuals, we calculate the Pareto Strength values for each individual $i$ in population $P$ as described in Zitzler et al. (2002):

$$S(i) = |\{j| j \in P \land i \succ j\}|$$ (5.10)

where $|\cdot|$ is the cardinality of a set, and $\succ$ is the dominance relation. Now we can calculate raw fitness values of each individual $i$ by summing the strengths of the individuals $j$ that dominate $i$:

$$f(i) = \sum_{j\in P, j\succ i} S(j)$$ (5.11)

From these formulae, it follows that if an individual is not dominated, its fitness value is 0. This is the single fitness metric that we will use to optimise our controllers.

This evaluation creates a single fitness value that reflects multiple fitness metrics, and we can straightforwardly plug these results into the standard NEAT algorithm\textsuperscript{vii} to complete NEAT-PS.

### 5.3.4 Experimental Set-Up

Our experimental set-up consists of two main parts: the vehicle simulator in which we tested our controllers, and NEAT-PS which evolves the vehicle controllers based on the feedback it obtains from the simulator. This setup is schematically depicted in Figure 5.18.

We used Movsim (Kesting, 2008), an open-source vehicle simulator that implements the Intelligent Driver Model (IDM) and MOBIL, which are respectively the longitudinal (determining minimum headway and consequent speed adjustments) model and lane changing model that form a benchmark in our experiments. Movsim is primarily a microscopic simulator of traffic, whereas we want to use higher-level actions as input for the vehicle (such as creating, joining and leaving a platoon). Therefore we had to extend the functionality of the simulator to

\textsuperscript{vii}We used the NEAT4j implementation, http://neat4j.sourceforge.net
provide these higher-level actions. In the original simulator, each vehicle can only use a single longitudinal controller and this controller decides on every timestep what the acceleration of the vehicle should be.

We extended the simulator so that each vehicle can use many controllers; a decision making process (such as a neural network) on a more abstract level selects the next action of the vehicle based on the current state of the vehicle. To execute the selected action the vehicle decides on the appropriate longitudinal controller for that moment. For example, when joining a platoon, the vehicle first has to drive towards the platoon using a specific longitudinal controller that facilitates this. Then, when the vehicle has approached the platoon, it may have to change to the correct lane. Other vehicles may have to make space for the joining vehicle. All these processes correspond with different longitudinal controllers. The original Movsim implementation directly links observable inputs to acceleration; we introduce an extra layer that chooses an action and a corresponding longitudinal model to execute.

The higher level actions implement realistic platooning behaviour, where vehicles on the highway drive very close to the preceding vehicles while copying the preceding vehicle’s speed and acceleration. The intricacies of the platooning control, such as sensing, information fusing and communication models are not taken into account in this simulation: these issues should be solved at the lower levels

Figure 5.18 – Graphical representation of our simulation loop. The NEAT algorithm generates a new population of controllers. Each member of a new generation of vehicle controllers is then evaluated by the vehicle simulation. The simulator assigns the fitness values (the Pareto strengths) to the controllers for NEAT to use in its parent and survivor operators. The loop is repeated for a preset number of generations.
Chapter 5. Preference-based Controller Building

Figure 5.19 – An example of a two-vehicle platoon. Note that the distance between the two platooning vehicles is much smaller than the distance between the non-platooning vehicles.

of the system. They are not within the scope of this work, and we assume these technologies to be working correctly and without error.

Figures 5.19 and 5.20 show two example screen shots of the simulation. Note that these are very zoomed-in screen shots and therefore only show a very small part of the total simulation.

5.3.4.1 Controller

By default, the vehicles in our simulation drive according to the Intelligent Driver Model (IDM), and the MOBIL lane changing model. Additionally, our vehicles can execute actions, each triggering other versions of longitudinal controllers.

Create Platoon If the vehicle is currently not part of a platoon, it creates a new platoon. The IDM longitudinal model remains active, and the lane changing model is turned off (in our simulation, platoons cannot change lanes). The

Figure 5.20 – A join scenario. The light-blue vehicle was already part of the platoon, and the dark blue vehicle wants to join. After the light-blue vehicle has made sufficient space, the blue vehicle will change lanes and be part of the platoon.
vehicle that creates the platoon automatically becomes the platoon leader – the foremost vehicle in the platoon.

**Join Platoon Front** If the vehicle is currently not part of a platoon, it tries to join a nearby platoon. This action checks if there is a platoon directly in front of the vehicle. If so, the vehicle drives up to the last vehicle in that platoon, and changes its longitudinal model to CACC, which copies the velocity and acceleration of the preceding vehicle.

**Join Platoon Side** This action joins a platoon that is directly next to the vehicle. First, the vehicle checks if there is a platoon on its side, and if so, the platoon makes way for the vehicle to join. As soon as there’s enough space for this, the vehicle switches lanes and joins the platoon by switching to the CACC longitudinal model.

**Leave Platoon** If the vehicle is part of a platoon, the vehicle leaves the platoon, and starts driving using the IDM and MOBIL models again. Immediately, more headway time is created. Additionally, if the vehicle that leaves the platoon was driving somewhere in the middle of the platoon, the platoon is split into two parts. If the leaving vehicle was platoon leader, this role is transferred to the second vehicle in the platoon. If the leaving vehicle was the only vehicle in the platoon, the platoon is disbanded.

**Change Lane** If possible, the vehicle changes lanes. Our simulation currently consists of two lanes, so this action does not need a parameter telling which lane to change to.

**Do Nothing** The vehicle longitudinal controller remains unchanged.

### 5.3.4.2 Neural Network

NEAT-PS evolves the neural network that selects from the six actions listed above. The network inputs and outputs are depicted in Figure 5.21, listing the following input variables for a vehicle $c$:

- Headway time of vehicle to preceding vehicle: The distance to the preceding vehicle divided by the velocity of $c$;
− Fraction of front platoon speed and desired velocity: The speed of the platoon in front of \( c \) (if there is one) divided by \( c \)'s desired velocity;

− Fraction of side platoon speed and desired velocity: The speed of the platoon on the side of \( c \) (if there is one) divided by \( c \)'s desired velocity;

− Fraction of desired velocity: The current velocity of \( c \) divided by its desired velocity;

− isPlatooning?: The current platooning status of \( c \);

− Headway time to preceding vehicle on other lane: the distance to the first preceding vehicle on the other lane divided by \( c \)'s velocity.

The input variables can easily be modified or extended with other variables, but it has to be kept in mind that the vehicle should be able to register these variables autonomously and on a real-time basis. The values of the input variables are normalised to \([0, 1]\) (except those inputs that are a fraction of the desired speed, those can be greater than 1) to make them more amenable as inputs for the neural network.

The six output variables select from the actions that we have defined in our simulation: the controller simply picks the action with the highest corresponding output value.

5.3.4.3 Evaluation Criteria

We used the following six objectives in our experiments:

**Speed** The first objective in this study is speed, simply measured as the time it takes a vehicle to complete the stretch of motorway in our simulation.

**Comfort: Lane Changes** As a measure of the comfort of a ride, we take the number of times a vehicle changes lanes during a ride, assuming that fewer lane changes indicates a more comfortable ride.

**Comfort: Vehicle Jerk** This is a measure of how ‘smooth’ the vehicle drives on the highway. It is the rate of change of acceleration or the derivative of acceleration with respect to time. We sum the absolute value of this measure for each vehicle \( c \) and each time step \( t \).
5.3. Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives

**Figure 5.21** – The input and output layers of the neural network controller of the vehicles. The inputs are observables from the simulation, and the outputs are the actions that need to be performed (if possible). The neural network layers in between the input and output layers are evolved by NEAT.

**Fuel Economy** We also measure the fuel used in the simulation using the model defined in Treiber and Kesting (2013). This model takes into account several vehicle characteristics, such as engine power and vehicle weight, for which we used default values, that are appropriate for regular commercially available vehicles.

**Failed Actions** The neural network decides which action to perform, but does not know beforehand if the action is possible. This objective gives the neural network feedback about the number of failed actions. The lower this number, the more appropriate actions the network chooses.

**Performance Consistency** This measure is a bit different than the others. It is not measured after a single run of the simulation, but after 10 runs of a single individual. We measure consistency by the mean Euclidean distance between these individuals. The lower this distance, the more consistent this controller performs across different runs.

To evaluate each controller, we simulate a 10 km, two-lane stretch of road with 3000 vehicles entering per hour, all running that particular neural network. Each vehicle has a desired speed of 30 m/s. Their initial speeds differ by a random amount up to 5 m/s from the desired speed.
Each vehicle evaluates the neural network every 5 seconds, and executes the action that the neural network proposes. The vehicle only executes the action if it is possible. For example, the vehicle will not change lanes if this would result in a crash. If an action is not executable, the vehicle will simply remain using the longitudinal controller it was already using, waiting another 5 seconds before re-evaluating the neural network.

We run this simulation for 10 minutes of simulation time (i.e. ca. 500 vehicles enter the stretch of road), after which we measure first 5 aforementioned objectives. After we evaluated an individual 10 times, we compute the sixth objective, which is the mean Euclidean distance between individuals. These measurements are then used to calculate the controller’s Pareto Strength as described above.

In this setting, we ran 50 repeats of 50 generations each with population size 50. From the 50 final generations that obtained from these runs, we extract the combined non-dominated solutions.

To assess the quality of the evolved neural net controllers, we also performed 10 repeats of a benchmark experiment where the vehicles are controlled using only the IDM behavioural model.

Table 5.3 summarises the experimental settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road length</td>
<td>10km</td>
</tr>
<tr>
<td>Vehicle inflow</td>
<td>3000veh/hr</td>
</tr>
<tr>
<td>Desired speed</td>
<td>approx. 30m/s</td>
</tr>
<tr>
<td>Max deceleration</td>
<td>-3m/s²</td>
</tr>
<tr>
<td>Simulation time</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Timestep size</td>
<td>0.20 seconds</td>
</tr>
<tr>
<td>Neural Network Evaluations</td>
<td>Every 5 seconds</td>
</tr>
<tr>
<td>Neural Network inputs</td>
<td>6</td>
</tr>
<tr>
<td>Neural Network outputs</td>
<td>6</td>
</tr>
<tr>
<td>Population Size</td>
<td>50</td>
</tr>
<tr>
<td>Fitness Evaluations per individual</td>
<td>10</td>
</tr>
<tr>
<td>Generations</td>
<td>50</td>
</tr>
<tr>
<td>Repeats</td>
<td>50</td>
</tr>
<tr>
<td>Benchmark runs</td>
<td>10</td>
</tr>
<tr>
<td>Validation runs</td>
<td>10</td>
</tr>
</tbody>
</table>
5.3. Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives

5.3.5 Results and Analysis

From all our experiments, we extracted the set of non-dominated individuals. It turns out that we have over 700 non-dominated individuals, aggregated from our 50 runs of 50 individuals. This high number is not surprising, because of the high-dimensionality of the data: it is very easy for any individual to be non-dominated, since the domination relation requires an individual to be better at every single objective.

Because of this very high number of non-dominated individuals, and the complexity of visualising high-dimensional data, we split the data in non-dominated individuals for each pair of objectives. This makes it easier to visualise and interpret the data. The downside is that this method results in more plots than we have space. Therefore, we show the results for one of the objective pairs (Travel time and # Lane Changes).

Figures 5.22 and 5.23 show these results. Figure 5.22 shows all the non-dominated individuals for all 6 objectives in grey, and in the top left chart, the non-dominated individuals for only those objectives are shown in black. We highlighted these individuals in the other charts as well, in blue. This gives insight in what happens in performance of the other objectives when you choose different controllers that are dominating in one particular area.

Several interesting properties can be derived from Figure 5.22. For example, we see that if we switch between these black controllers, it has a big impact on fuel consumption. If you want a faster controller, your fuel consumption goes up. However, in terms of failed actions and vehicle jerk, this has very little influence. The blue dots in the corresponding charts lie very closely to each other.

It is interesting to note that there are particularly many failed actions in these controllers. This practically means that the controller performs almost no actions at all, which effectively means that the IDM controller is activated in these cases. The number of failed actions is therefore probably not the best objective function.

In Figure 5.23, we show the results of the validation runs and the benchmark runs. The original non-dominated set is shown in blue, the validation runs in red and the benchmark runs in black. We see that the red validation runs lie neatly above their original counterparts, which means that consistency among runs is good. For five of the six objectives, we show the validation runs as the interquartile range of these runs. The validation of the consistency between runs
is only a single point, since this metric is based on 10 runs, which is exactly the number of validation runs we ran.

We also compared our results with the IDM controller. We measured all objectives with this controller, except for the failed actions objective, since that objective is only applicable with our neural network controllers. We see that the consistency of this model is very high. In terms of lane changes, it performs much worse than our evolved controllers. However, it consumes little fuel, comparable with the best individual from our subset of non-dominated individuals. In the travel time objective, it performs as good as the slowest non-dominated individual. The vehicle jerk is comparable to that of all the non-dominated controllers.

In earlier work (van Willigen et al., 2013c), we did similar work, but only with 2 objectives. These objectives are the ones that we also analysed in detail above, travel time and lane changes. In Figure 5.24, we compare the results. The blue line and red crosses are the results from this study, and the green line and grey crosses are the results from the earlier work. We see that our current method is slightly worse than if you only use 2 objectives. This is not surprising, since there are simply many more objectives to optimise. In fact, the current results are only slightly worse.

We also see that the density of non-dominated solutions is less in our current results. This can be explained by the fact that we chose these solutions from the global non-dominated set of solutions, which is already a subset of the total population. In the previous work, we chose the solutions from the total population.

5.3.6 Conclusions and Future Work

We described our vision of a future in which vehicles on highways can drive autonomously using Cooperative Adaptive Cruise Control. With CACC technology, vehicles are able to form trains of vehicles that drive very close to one another, also known as platoons. We performed experiments that show that multi-objective evolutionary algorithms can evolve a number of controllers that embody a variety of prioritisations of ride qualities like speed, comfort and fuel economy. Such a combination of controllers will, in our vision, allow the driver to emphasise different qualities based on her preference and needs in real time.
5.3. Fast, Comfortable or Economical: Evolving Platooning Strategies with Many Objectives

The experiments described in this study are based on six objectives: speed, two comfort measures (lane changing and vehicle jerk), fuel consumption, failed actions and consistency.

The results of the experiments show that the method still holds when substantially increasing the number of objectives. It performs marginally worse, and the resulting non-dominated set of controllers is slightly smaller than the results we found in previous work (van Willigen et al., 2013c), but this was expected and the results are not significantly worse.

In this section, there was only space to analyse the interaction between the non-dominated set of one pair of objectives and the rest of the objectives. We obtained good insights about the relations between certain objectives. The results of all the other pairs of objectives can be found on-line on the author’s website\(^{\text{viii}}\).

The evolved controllers outperform the benchmark IDM controller substantially. The IDM is a widely used model of human driver behaviour; we conclude

\(^{\text{viii}}\)http://www.researchgate.net/profile/WH_Van_Willigen

![Figure 5.22](image-url)
that automatic driving as we envisage will substantially increase travel speed as well as ride comfort.

The concept of evolving many non-dominated controllers represents a significant improvement from the current practice of optimising a weighted average of

Figure 5.23 – Validation results for the set of non-dominated individuals from the objective combination Travel time and # Lane Changes, and the performance of the IDM benchmark controller. The original non-dominated points are also included.

Figure 5.24 – Validation results for the set of non-dominated individuals from the objective combination Travel time and # Lane Changes. The original non-dominated points are also included (in blue).
objectives to obtain a single controller that prioritises ride qualities as the developer deems appropriate. The consequent freedom of choice for drivers to select controller behaviour as they see fit will greatly benefit acceptance of CACC technology.

These results open up possibilities for substantial further research. One track of future work includes tests with heterogeneous vehicles. In our current work, we tested a particular neural network by letting all vehicles in a simulation use the same network. The question remains whether these networks remain their performance when other vehicles have completely different controllers or even human drivers on board.

The fuel consumption model that we used in this work is only based on characteristics of individual vehicles alone, and does not take vehicle drag into account. When platooning, this becomes more important, as platooning vehicles will consume less fuel when driving closely behind another vehicle. We plan to include this in our simulations as well.

Another track for future research is to drastically increase the number of individuals in the population and the number of generations. We hypothesise that this improves the quality of the non-dominated individuals as well.
5.4 The Fast and the Comfortable: Evolving Preference-Based Controllers for Autonomous Vehicles

The research in this section is inspired by a vision of intelligent vehicles that autonomously move along motorways: they join and leave trains of vehicles (platoons), overtake other vehicles, etc. We propose a multi-objective evolutionary algorithm that evolves high-level controllers for such intelligent vehicles. The algorithm yields a set of solutions that each embody their own prioritisation of various user requirements such as speed, comfort or fuel economy. We show that it is possible to evolve a set of vehicle controllers that correspond with different prioritisations of user preferences, giving the driver, on the road, the power to decide which preferences to emphasise. We show that it is in principle possible and beneficial to employ multi-objective evolution to develop such sets of controllers. We find that increasing the number of objectives can result in performance loss on some objectives, but this seems an effect of particular combinations of objectives rather than of adding objectives per se.

5.4.1 Introduction

In Europe’s major cities, the average driver loses as much as 60 to 70 hours per year in traffic jams,\(^\text{1}\) the cause of tremendous environmental and socio-economical impact. Vehicle manufacturers are introducing Adaptive Cruise Control (ACC) to help combat this issue. ACC allows the driver to set the desired distance to the preceding vehicle. The ACC system measures the vehicle’s relative distance and velocity through radar and adjusts speed accordingly. Such automated mainte-

---


Section 5.4 is under review as:

nance of distance is hoped to increase the uniformity of traffic flows and so better utilise road capacity. Cooperative Adaptive Cruise Control (CACC) is an important development in recent research on Intelligent Transportation Systems (ITS) that takes cruise control to the next level by adding direct communication between vehicles. With vehicles capable of directly communicating their current velocity, position and acceleration, CACC removes some of the uncertainty and delays that relying only on radar measurements implies. Directly communicating accurate state information allows vehicles to drive much closer to each other without compromising safety.

CACC affords the possibility of platooning: trains of vehicles driving very close to each other at (near) equal speed, and therein lies one of the big selling points of CACC: traffic throughput is increased, reducing traffic jams and increasing driver safety and comfort. Furthermore, this method of reducing traffic jams requires little or no infrastructural investments: it is cheap compared to increasing highway capacity by adding extra lanes. Also, CACC technology can be developed without governmental commitment.

CACC will become available in the near future: road tests are rapidly becoming everyday occurrences Ploeg et al. (2011b); Bu et al. (2010). Vehicles will be equipped with a switch to turn the CACC mode on and off; with the system turned on, the vehicle will, for instance, join the platoon ahead and (almost) automatically drive along in the platoon until the driver decides to take matters into her own hands and turn CACC off again.

Such CACC systems fall short of our vision of autonomous vehicles: we envision a system where the car controller decides whether to join a platoon, drive alone, overtake, switch to another platoon, etcetera. The controllers choose actions (e.g., joining, leaving or creating platoons, changing lanes or simply driving on) based on a number of input variables such as acceleration, velocity, and distance to preceding vehicle.

This paper highlights an aspect of autonomous vehicle control that, in our view, will be crucial to ensure adoption of this technology by the public: the driver must be able to select desired driving behaviour at run-time. Sometimes, the driver will want to drive fast, while at other times, the driver may prefer fuel economy and care less about speed. Ride comfort is another consideration – it can be measured in terms of the number of lane changes, changes in velocity, etc. This aspect of dynamic preferences is often disregarded in current research.
into controller development: driver preferences are typically taken into account by simply condensing them in a single metric (e.g. a weighted sum), and a single controller is then developed that optimises this metric.

We propose to offer this possibility of real-time switching between controller behaviour by employing not a single controller, but by having multiple controllers, each with its own unique prioritisation between these user preferences: some drive fast at the cost of fuel economy, others emphasise comfort, etc. The driver can then indicate what kind of driving behaviour she currently requires and the system selects the most appropriate controller from the set of available controllers: setting preferences corresponds with activating the vehicle controller that best approximates the current prioritisation of preferences.

Multi-objective evolutionary algorithms typically allow for optimisation without having to summarise the separate preference metrics into a single value. They explore various prioritisations of the objectives and yield multiple controllers that each reflect their own prioritisation of preference measures. Such algorithms therefore are well-suited to deliver the sets of controllers that we propose to employ. Thus, we see the benefit of evolutionary multi-objective optimisation of autonomous vehicle controllers not so much in better control (although that would of course not be disregarded), but more in the fact that it would yield multiple controllers with varying emphasis on the different user preferences and so allow the user to select and change the appropriate vehicle behaviour while driving.

We implemented this idea by combining NEAT, a well-known algorithm for neuro-evolution Stanley and Miikkulainen (2002) with the Pareto Strength method as found in the SPEA2 evolutionary algorithm Zitzler et al. (2002). In van Willigen et al. (2013c), we provided a proof-of-concept of this NEAT-Pareto Strength (NEAT-PS) combination to yield a useful set of controllers that embodied varying prioritisations of speed and comfort (expressed as the number of lane changes).

In this paper, we explore the possibility of extending our approach to six objectives. Additional objectives include fuel economy, an additional comfort measure, the number of failed actions and the consistency of a controller. We pose the following research questions:

1. Does extending the number of objectives impose a performance penalty on the original two objectives?
2. Is an observed performance decrease an inescapable consequence of adding objectives or is it (partly) caused by incompatibilities between particular objectives?

We conduct a number of simulation experiments in which we evolve controllers to optimise various combinations of objectives to answer these questions.

The contribution of this research is two-fold: we combine two techniques from different corners of evolutionary computation, and we apply these particular techniques in the automated driving domain.

5.4.2 Related Work

CACC is a well-established research field, and in this paper we assume that the vehicles have CACC capabilities so that they can form and maintain platoons. Van Arem et al. (2006) describe the effect of CACC on traffic flow and conclude that, when the penetration level of CACC-equipped vehicles is high enough (> 60%), traffic stability and throughput is improved. In Yang et al. (2004), a communication protocol is proposed in order to make a cooperative collision warning system on highways.

There has not been much work yet on the development of platooning strategies, where the vehicles have to decide while driving if and when they should be platooning. In the SARTRE project (Safe Road Trains for the Environment), as described in Bergenhem et al. (2010), platoons are defined as trains of road vehicles of which the front one is a trained platoon driver. This project has implemented basic platooning strategies, also in real vehicles, but these strategies are only defined at the action level, where for example the join and leave actions are defined, but not at a higher level, where a strategy for driving on the highway is defined.

Evolutionary Algorithms (EAs) encompass a wide range of population-based meta-heuristics typically used for optimisation of solutions to problems ranging from routing to robot control. EAs are inspired by the process of natural evolution, optimising populations of solutions in a trial-and-error process of selection and modification. Eiben and Smith (2003) and Floreano and Mattiussi (2008) are extensive overviews of EAs and similarly nature-inspired techniques.

To optimise multiple objectives, most traditional optimisation methods must condense these objectives into a single value, typically using a weighted sum approach. The weights determine the relative priority of the objectives as defined
by the user of the optimiser, at optimisation time. Because EAs consider whole populations of solutions simultaneously, they have a great benefit when optimising multiple objectives: they can explore multiple prioritisations of the objectives simultaneously. Employing selection criteria that promote a thorough exploration of the Pareto front, multi-objective EC algorithms result in a set of solutions, each embodying a different prioritisation of the objectives Deb (2001).

Of particular interest for this paper is the field of Evolutionary Robotics, where EAs are employed to develop robot controllers and indeed morphologies. These controllers—functions that transform a sensory inputs into motor commands—are often represented as artificial neural networks, but with the network details determined evolutionarily rather than through backpropagation or similar techniques. See Bongard (2013) for a recent overview of this field.

In a rare application of EAs in the ITS domain, Dovgan et al. (2011, 2012) turned to evolution for the development of rule-based driving strategies. They employ a multi-objective algorithm called NSGA-II to develop controllers for individual autonomous cars that optimise travel time and fuel economy. Other vehicles or platooning are not taken into account in this work: the controllers only consider the vehicle’s own state and that of the environment.

EA applications are uncommon in ITS research, although some researchers have tried to couple the domains in different application areas, such as path planning by Lin et al. (2009) and sensor placement on autonomous vehicles in Martinoli et al. (2002).

### 5.4.3 Controller

The vehicles in our simulation employ the Intelligent Driver Model (IDM), and the MOBIL lane changing model as base behaviour—simply going straight ahead as close as possible to the desired speed, maintaining a safe distance and overtaking as necessary. The vehicle controller can override this base behaviour to execute higher-level actions, implemented as other longitudinal controllers. The lower-level control mechanisms of the vehicles are assumed to be working correctly and safely, and are able to deal with uncertain behaviour of sensors and other users on the road. The development of these lower-level controllers is a different field of research altogether, and falls outside the scope of this section. We refer the
interested reader to Althoff and Dolan (2011) and Ali et al. (2013), about collision detection given uncertain behaviour of other traffic participants.

The vehicles in our model can perform the following actions:

- **Create Platoon** If the vehicle is currently not part of a platoon, it creates a new platoon. The IDM longitudinal model remains active, and the lane changing model is turned off, so that platoons cannot change lanes. The vehicle that creates the platoon automatically becomes the platoon leader – the front vehicle in the platoon.

- **Join Platoon Front** If the vehicle is currently not part of a platoon, it tries to join a nearby platoon. This action checks if there is a platoon directly in front of the vehicle. If so, the vehicle drives up to the last vehicle in that platoon, and changes its longitudinal model to CACC, which directly copies the velocity and acceleration of the preceding vehicle in the platoon.

- **Join Platoon Side** This action joins a platoon that is directly next to the vehicle. First, the vehicle checks if there is a platoon on its side, and if so, the platoon makes way for the vehicle to join. As soon as there’s enough space, the vehicle switches lanes, joins the platoon and switches to the CACC longitudinal model.

- **Leave Platoon** If the vehicle is part of a platoon, the vehicle leaves the platoon, and starts driving using the IDM and MOBIL models again. As a result, more headway time is created. Additionally, if the vehicle that leaves the platoon was driving somewhere in the middle of the platoon, the platoon is split into two separate platoons. If the leaving vehicle was platoon leader, this role is transferred to the second vehicle in the platoon. If the leaving vehicle was the only vehicle in the platoon, the platoon is disbanded.

- **Change Lane** If possible, the vehicle changes lanes. Our simulation currently consists of two lanes, so this action does not need a parameter telling which lane to change to.

- **Do Nothing** The vehicle longitudinal and lateral controllers remain unchanged.
If an action fails (for example if there is no room to change lanes, or if there is no platoon to join), the longitudinal and lateral controllers of the vehicle remain unchanged.

### 5.4.3.1 Neural Network

In our experiments, we use an artificial neural network as a controller that selects the high-level subcontrollers mentioned above. In (evolutionary) robot control, neural networks are often the technique of choice for sensor-actuator mappings. In our case, the network creates a mapping between sensor data (the inputs) and high-level vehicle actions (the outputs). The output nodes of the neural network select the higher-level actions based on the following inputs:

- **Headway time of vehicle to preceding vehicle** The distance to the preceding vehicle divided by the velocity of vehicle \( c \);

- **Fraction of front platoon speed and desired velocity** The speed of the platoon in front of \( c \) (if there is one) divided by vehicle \( c \)'s desired velocity;

- **Fraction of side platoon speed and desired velocity** The speed of the platoon on the side of \( c \) (if there is one) divided by vehicle \( c \)'s desired velocity;

- **Fraction of desired velocity** The current velocity of vehicle \( c \) divided by its desired velocity;

- **isPlatooning?** The current platooning status of vehicle \( c \);

- **Headway time to preceding vehicle on other lane** the distance to the first preceding vehicle on the other lane divided by vehicle \( c \)'s velocity.

These input variables can easily be modified or extended to reflect additional sensors, but it has to be kept in mind that the vehicle should be able to register these variables autonomously and on a real-time basis. The values of the input variables are normalised to \([0, 1]\) (except those inputs that are a fraction of the desired speed, those can be greater than 1) to make them more amenable as inputs for the neural network.

The six output variables select from the actions that we have defined in our simulation: the controller picks the action with the highest corresponding output value. Figure 5.25 schematically depicts this arrangement.
Figure 5.25 – The input and output layers of the neural network controller of the vehicles. The inputs are observables from the simulation, and the outputs are the actions that need to be performed (if possible). The neural network layers in between the input and output layers are evolved by NEAT.

5.4.3.2 Neuro-evolution

We develop the neural networks that provide high-level control using neuro-evolution. Neuro-evolution employs population-based population-based meta-heuristics inspired by natural evolution (evolutionary algorithms) to train artificial neural networks. The evolutionary algorithm comprises of a generate-and-test loop that proposes candidate neural networks, evaluates their performance at some task (in our case, selecting appropriate low-level actions), selects better performing candidates and develops new candidates from those through variation operators (mutation and/or recombination). As this process iterates, it develops increasingly well-performing neural networks. It should be noted that neuro-evolution, and meta-heuristics in general, do not guarantee convergence to the global optimum, because of the huge search space, the possibility of many local optima and the iterative nature of these algorithms.

The internal layout of the neural network –the composition of the hidden layer(s)– of the neural network has to be specified, and designing this is a non-trivial task. Many neuro-evolution algorithms optimise only the weights between nodes in a given neural network. The neuro-evolutionary technique that we employ in this work seeks not only to optimise weights of a neural network, but the
internal layout of these networks as well. This means that we conveniently don’t have to decide on a topology for the neural network beforehand. This also allows for the algorithm to propose functionally quite different neural networks.

In our work, we use NEAT, a popular state-of-the-art neuro-evolutionary algorithm Stanley and Miikkulainen (2002). NEAT employs a clever method of specification of the neural network topologies, taking into account specific sub-structures within the topology. The neural networks compete primarily with individuals within their own niches instead of with individuals of the entire population. This technique makes it easier to protect innovations within the topologies.

The following section describes how we evaluate the performance of the generated neural networks.

5.4.3.3 Multi-Objective Optimisation

To achieve our vision of optimising multiple user preferences, it is essential to take multiple objective functions into account. The original NEAT algorithm, however, does not support multiple objectives. To extend the NEAT algorithm and so allow for multi-objective optimisation, we incorporated the Pareto Strength approach from SPEA2 into NEAT to create NEAT-PS. The Pareto Strength approach is easily integrated into existing EAs because it computes a single fitness value for each individual candidate solution, based on the values of all objective functions.

Although it does compute a single value to serve as basis for selection in the EA, Pareto Strength is in no way like a weighted sum over all the objectives. The basis of the Pareto Strength evaluation is the domination relation:

\[ i \succ j \iff \forall o \in O : o_i \succeq o_j \land i \neq j \quad (5.12) \]

In this formula, the \( \succeq \) relation between objectives only specifies that objective \( o_i \) is equal or better than objective \( o_j \), without saying whether the objective is maximised or minimised. In our specific experiments, we are minimising all objectives.

After calculating each objective function’s value for all individuals, we calculate the Pareto Strength values for each individual \( i \) in population \( P \) as described in Zitzler et al. (2002):

\[ S(i) = |\{j| j \in P \land i \succ j\}| \quad (5.13) \]
where \(|·|\) is the cardinality of a set, and \(\succ\) is the dominance relation. Now we can calculate raw fitness values of each individual \(i\) by summing the strengths of the individuals \(j\) that dominate \(i\):

\[
f(i) = \sum_{j \in P, j \succ i} S(j)
\]  

(5.14)

From these formulae, it follows that if an individual is not dominated, its fitness value is 0. This is the single fitness metric that we will use to optimise our controllers.

This evaluation creates a single fitness value that reflects multiple fitness metrics, and we can straightforwardly plug these results into the standard NEAT algorithm.\(^x\)

The set of solutions that are not dominated by any other solution—the best found prioritisations of the objectives—is called the non-dominated set or Pareto front. This is the set that we propose to implement as controllers that the user can choose from by setting preferences like “low fuel consumption, comfortable driving style”.

It should be noted that our high-level controllers were not checked and verified in terms of uncertain behaviour and stability: when many cars on the road would constantly change their high-level controller (that corresponds with a certain driving style), this could result in unstable behaviour.

### 5.4.4 Experimental Set-Up

Our experimental set-up consists of two main parts: the vehicle simulator in which we tested our controllers, and NEAT-PS which evolves the vehicle controllers based on the feedback it obtains from the simulator. This setup is schematically depicted in Figure 5.26.

Movsim(Kesting, 2008) is an open-source vehicle simulator that implements the Intelligent Driver Model (IDM) and MOBIL, which are respectively the longitudinal (determining minimum headway and consequent speed adjustments) model and lane changing model that define base vehicle behaviour. Movsim is primarily a microscopic simulator of traffic, whereas we want to use higher-level actions as input for the vehicle (such as creating, joining and leaving a platoon). Therefore we

\(^x\)We based our implementation on NEAT\(_4\)j http://neat4j.sourceforge.net.
Chapter 5. Preference-based Controller Building

**Figure 5.26** – The simulation loop. The EA (NEAT) generates a new population of controllers. Each member of a new generation of vehicle controllers is then evaluated by the vehicle simulation. The simulator assigns the fitness values (the Pareto strengths) to the controllers for the EA to use in its parent and survivor operators. The loop is repeated for a preset number of generations.

**Figure 5.27** – The five-layered architecture of our vehicle controllers. In the original simulation settings, the three layers in the middle are missing, thus calculating the acceleration directly based on the input variables.

extended the simulator to provide these higher-level actions. In the original simulator, each vehicle can only use a single longitudinal controller and this controller decides on every timestep what the acceleration of the vehicle should be.

We extended this architecture so that each vehicle can use many controllers; a decision making process (in our case, a neural network) on a more abstract level selects the next action of the vehicle based on the current state of the vehicle. To execute the selected action the vehicle selects the appropriate longitudinal controller for the current action. For example, when joining a platoon, the vehicle first has to drive towards the platoon using a specific longitudinal controller that facilitates this. Then, when the vehicle has approached the platoon, it may have to change
to the correct lane. Other vehicles may have to make space for the joining vehicle. All these processes correspond with different longitudinal controllers. The original Movsim implementation directly links observable inputs to acceleration; we introduce an extra layer that chooses an action and a corresponding longitudinal model to execute, as displayed in figure 5.27.

The higher level actions implement realistic platooning behaviour, where vehicles on the highway drive very close to the preceding vehicles while copying the preceding vehicle’s speed and acceleration. The intricacies of the platooning control, such as sensing, information fusing and communication models are not taken into account in this simulation: these issues are assumed to be solved at the lower levels of the system.

Figures 5.28 and 5.29 show two partial example screen shots of the simulation. Note that these are very zoomed-in screen shots and therefore only show a very small part of the total simulation.

Figure 5.28 – An example of a two-vehicle platoon. Note that the distance between the two platooning vehicles is much smaller than the distance between the non-platooning vehicles.

![Figure 5.28](image)

Figure 5.29 – A join scenario. The light-blue vehicle was already part of the platoon, and the dark blue vehicle wants to join. After the light-blue vehicle has made sufficient space, the blue vehicle will change lanes and be part of the platoon.

![Figure 5.29](image)
5.4.4.1 Objectives

We did experiments in which we evolved controllers based on 2, 3, 4, 5 or 6 objectives. The objective functions that we used are the following:

1. **Travel Time** The first objective is speed, simply measured by the time it takes a vehicle to complete the stretch of highway in our simulation.

2. **Comfort: Lane Changes** As a measure of the comfort of a ride, we take the number of times a vehicle changes lanes during a ride, assuming that fewer lane changes indicates a more comfortable ride.

3. **Fuel Economy** We also measure the fuel used in the simulation using the model defined in Treiber and Kesting (2013). This model takes into account several vehicle characteristics, such as engine power and vehicle weight, for which we used default values, that are appropriate for regular commercially available vehicles.

4. **Comfort: Vehicle Jerk** This is a measure of how ‘smooth’ the vehicle drives on the highway. It is the rate of change of acceleration, or the derivative of acceleration with respect to time. We sum the absolute value of this measure for each vehicle $c$ and each time step $t$.

5. **Failed Actions** The neural network decides which action to perform, but does not know beforehand if an action is possible. This objective gives the neural network feedback about the number of failed actions. The lower this number, the more appropriate actions the network chooses.

6. **Performance Consistency** This measure is a bit different than the others. It is not measured after a single run of the simulation, but after 10 runs of a single individual. We measure consistency by the mean Euclidean distance between these individuals. The lower this distance, the more consistent this controller performs across different runs.

Note that of these objectives, numbers 5 and 6 are not user preferences, but these objectives say something about the performance of the underlying controller. This means that these objectives will not be visible for the user (e.g. as a dial on the dashboard). We have included these objectives in this study because we do regard them as important quality parameters of the controller.
In earlier work, we investigated a two-objective scenario with *travel time* and *number of lane changes* as the objectives van Willigen et al. (2013b). The current study adds all possible permutations of the other four objectives to assess the impact of adding objectives.

To evaluate a controller, we simulate a 10 km, two-lane stretch of road with 3000 vehicles entering per hour, all running that particular neural network. Each vehicle has a desired speed of 30 m/s. Their initial speeds differ by a random amount up to 5 m/s from the desired speed.

Each vehicle evaluates the neural network every 5 (simulation) seconds, and executes the action that the neural network proposes. The vehicle only executes the action if it is possible. For example, the vehicle will not change lanes if this would result in a crash. If an action is not executable, the vehicle will simply remain using the longitudinal controller it was already using, waiting another 5 seconds before re-evaluating the neural network.

We run this simulation for 10 minutes of simulation time (i.e. ca. 500 vehicles enter the stretch of road), after which we measure the values of the current objectives for all vehicles that completed the 10 km stretch. These measurements are then used to calculate the controller’s Pareto Strength as described above.

In this setting, each experiment consists of 50 repeats of 50 generations each with population size 50. From the 50 final generations that obtained from these runs, we extract the combined non-dominated solutions.

To assess the quality of the evolved neural net controllers, we also performed 10 repeats of a benchmark experiment where the vehicles are controlled using only the IDM behavioural model.

Table 5.4 summarises the experimental settings.

### 5.4.5 Results and Analysis

Figure 5.30 shows two Pareto fronts for only the two objectives (*travel time* and *number of lane changes*) that were considered in our original case study van Willigen et al. (2013c). The green line is the front resulting from runs with only these original objectives. The blue front denotes the result of runs that consider all the objectives listed in section 5.4.4.1. Plainly, the second front contains considerably worse solutions than the original front (note that both objectives must be minimised). This indicates that there is a substantial price to pay in terms of solu-
Table 5.4 – Experimental Setup

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Road length</td>
<td>10km</td>
</tr>
<tr>
<td>Vehicle inflow</td>
<td>3000veh/hr</td>
</tr>
<tr>
<td>Desired speed</td>
<td>approx. 30m/s</td>
</tr>
<tr>
<td>Max deceleration</td>
<td>-3m/s²</td>
</tr>
<tr>
<td>Simulation time</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Timestep size</td>
<td>0.20 seconds</td>
</tr>
<tr>
<td>Neural Network Evaluations</td>
<td>Every 5 seconds</td>
</tr>
<tr>
<td>Neural Network inputs</td>
<td>6</td>
</tr>
<tr>
<td>Neural Network outputs</td>
<td>6</td>
</tr>
<tr>
<td>Population Size</td>
<td>50</td>
</tr>
<tr>
<td>Fitness Evaluations per individual</td>
<td>10</td>
</tr>
<tr>
<td>Generations</td>
<td>50</td>
</tr>
<tr>
<td>Repeats</td>
<td>50</td>
</tr>
<tr>
<td>Benchmark runs</td>
<td>10</td>
</tr>
<tr>
<td>Validation runs</td>
<td>10</td>
</tr>
</tbody>
</table>

Now that we know that there is a performance penalty for adding the objectives listed above, the question is whether this penalty is an inescapable consequence of adding objectives or whether it is (also) caused by incompatibilities between

![Image](image.png)

**Figure 5.30** – Resulting Pareto front with the original 2 and with all 6 objectives. The 2-objective front contains substantially better controllers.
particular objectives: does adding objectives always incur a penalty or is this only the case for particular combinations of objectives?

To investigate this, we conducted runs with various combinations of objectives and we want to compare the resulting Pareto fronts. To be able to quantify the difference in performance for the solutions on two Pareto fronts, we consider the area under the curves defined by the Pareto fronts. Because we consider objectives that must be minimised, the smaller this area, the better the solutions on the Pareto front. The ratio between the area under the original Pareto front and under the Pareto front with additional objectives quantifies the performance loss (or gain): if the ratio is close to 1.0, there is little difference, if the ratio is much smaller, there is a substantial loss of performance. Figure 5.31 elucidates graphically how this ratio is determined.

Comparing the original 2 and the full set of 6 objectives in this way yields a ratio of 51.0% – the normalised area under the 6-objective front is twice that under

![Diagram](image_url)

(a) Two non-dominated fronts to compare. The fitness axes range between 0 and 9.

(b) Normalise the fitness values between 0 and 1, and limit the area at at the worst found value of each objective (the horizontal and vertical gray lines).

(c) Extend each front as needed to include the boundary values.

(d) With the areas defined, the ratio between the two can be calculated.

**Figure 5.31** – Comparing two non-dominated fronts through the normalised area under the fronts.
Table 5.5 – Comparison of Pareto fronts for original the 2-objective experiment and experiments with added objectives. $n$ denotes the number of objectives used in the comparing experiment. The numbers in the Objectives column refer to the objectives as described in section 5.4.4.1.

<table>
<thead>
<tr>
<th>$n$</th>
<th>Objectives</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1,2,3</td>
<td>88.1%</td>
</tr>
<tr>
<td>3</td>
<td>1,2,4</td>
<td>107.6%</td>
</tr>
<tr>
<td>3</td>
<td>1,2,5</td>
<td>129.0%</td>
</tr>
<tr>
<td>3</td>
<td>1,2,6</td>
<td>100.4%</td>
</tr>
<tr>
<td>4</td>
<td>1,2,3,4</td>
<td>97.3%</td>
</tr>
<tr>
<td>4</td>
<td>1,2,3,5</td>
<td>87.6%</td>
</tr>
<tr>
<td>4</td>
<td>1,2,3,6</td>
<td>87.9%</td>
</tr>
<tr>
<td>4</td>
<td>1,2,4,5</td>
<td>99.7%</td>
</tr>
<tr>
<td>4</td>
<td>1,2,4,6</td>
<td>77.8%</td>
</tr>
<tr>
<td>4</td>
<td>1,2,5,6</td>
<td>92.3%</td>
</tr>
<tr>
<td>5</td>
<td>1,2,3,4,5</td>
<td>90.7%</td>
</tr>
<tr>
<td>5</td>
<td>1,2,3,4,6</td>
<td>77.0%</td>
</tr>
<tr>
<td>5</td>
<td>1,2,3,5,6</td>
<td>59.7%</td>
</tr>
<tr>
<td>5</td>
<td>1,2,4,5,6</td>
<td>89.2%</td>
</tr>
<tr>
<td>6</td>
<td>1,2,3,4,5,6</td>
<td>51.0%</td>
</tr>
</tbody>
</table>

the original front. Table 5.5 lists the ratios for all combinations of objectives that we tried, with the original objectives 1 and 2 always included. The Pareto fronts for these experiments are shown in figures 5.32–5.34.

Strikingly, the results show that adding a third objective doesn’t necessarily decrease the performance compared to the original experiment. In fact, three out of the four 3-objective experiments perform better than the 2-objective experiment (denoted by the numbers higher than 100% in the table). Adding the Failed Actions objective (5) results in the biggest increase in performance, and only the Fuel Economy (3) objective decreases performance.

Even adding a fourth objective does not necessarily result in much worse performance on the two original objectives. There is a clear interaction here: adding the Performance Consistency (6) objective by itself had a positive impact, but combining them with the Comfort: Vehicle Jerk objective (4) incurs a large drop in performance for the original objectives.

Going to 5 objectives, we see that Fuel Economy (3), Performance Consistency (6) with Comfort: Vehicle Jerk (4) or Failed Actions (5) provide a poisonous combination,
which is even stronger in the 6-objective case. We still see a quite acceptable 90.7% for one combination of 5 objectives, though.

These results indicate that considering many objectives (allowing for a finely grained constellation of user preferences) may cause somewhat worse performance across the board. The impact of ‘incompatible’ combinations of objectives is much higher, though.

Up to this point, we considered incompatibilities between objectives in terms of the impact on the quality of controllers that NEAT-PS delivers. Now, we consider the interaction between objectives when it comes to choosing a controller that prioritises between certain user preferences.

Consider a scenario where the user can modify four preferences for autonomous driving behaviour: travel time, lane changes, fuel economy and cumulative jerk. By setting preferences, the user causes the selection of one of the controllers on the Pareto front shown in figure 5.35.

**Figure 5.32** – Resulting fronts with 3 objectives
Suppose the user initially decides that fuel economy is the only important objective; in that case a controller is selected that performs best on that particular objective – this controller is highlighted as number 1 in figure 5.35.

After a while, the user requests a more comfortable driving style by setting the lane changes objective to equal importance. This causes the vehicle to switch to
controller number 2, which performs substantially better on lane changes without paying too much in terms of fuel economy.

Next, the user decides to add the travel time objective, causing the selection of controller number 3. This has an obvious negative influence on fuel economy – the faster the vehicle moves, the more fuel it uses, but the effect in terms of lane changes is limited.

Finally, the user wants to increase drive comfort once more and selects the fourth objective, cumulative jerk. Now, controller 4 is selected. It performs worse in terms of travel time, but cumulative jerk improves, as well as lane changes and fuel economy.

This brief scenario highlights the strength of the multi-objective approach: controllers are chosen from the total non-dominated set of individuals, each with its own unique prioritisation of objectives. The density of the Pareto front ensures that changes in user priorities result in a smooth transition in driving behaviour and that combining incompatible preferences such as speed and economy results
in a graceful degradation of either objective. Compatible objectives are combined effortlessly.

Figure 5.35 – Pairwise plots for objectives travel time, lane changes, fuel economy and vehicle jerk. Each point indicates the performance for two objectives of one of the controllers on the Pareto front. For instance, the top-left graph plots lane changes vs travel time. Four controllers are highlighted in black; these represent four different combinations of user preferences as described in the text.

5.4.6 Conclusion

We described our vision of a future in which vehicles on highways can drive autonomously, able to create, join and leave platoons – trains of vehicles that drive very close to one another.

Our experiments show that multi-objective neuro-evolution can effectively combine various objectives, resulting in a set of artificial neural network-based controllers that each embody a particular prioritisation of ride qualities like speed, comfort and fuel economy. This set is dense enough to allow for fine-grained selection of controllers to match user-defined preferences. Consequently, chang-
ing preferences results in smooth changes in ride quality and incompatibilities between preferences can be resolved in a balanced manner.

Adding objectives can incur a slight degradation in terms of quality of controllers on originally considered objectives, although in some cases, performance on the original objectives increases when adding an objective. It seems that substantial performance loss is not the result of considering multiple objectives per se (at least not up to five or six objectives), but that some combinations of objectives hamper the developmental process and result in suboptimal solutions.

We considered permutations of up to six objectives: speed, two comfort measures (number of lane changes and vehicle jerk), fuel consumption, failed actions and controller consistency. These six objectives are illustrative for the kind of preferences that one may consider, but by no means the only objectives that apply. Other objectives can be considered, and future work should certainly look into this, including the impact of combining them in a multi-objective setting. An important finding of this study is that one should combine objective functions carefully: some combinations of objectives disrupt the evolutionary process, while others strengthen it.

The concept of evolving many non-dominated controllers represents a significant improvement over the current practice of optimising a weighted average of objectives to obtain a single controller that prioritises ride qualities as the developer deems appropriate. The consequent freedom of choice for drivers to select controller behaviour as they see fit will greatly benefit acceptance of autonomous vehicle technology. We considered fully autonomous vehicles in our research (although we do not consider the problem of providing reliable low-level control), but our approach can equally well be applied when developing controllers that govern more limited aspects of driving, for instance to maintain distance in ACC scenarios.

Future work should consider more sophisticated simulation scenarios where many different controllers, maybe even mixed with models of human drivers are tested together in the same simulation. Similarly more realistic simulators, ultimately even real vehicles, will strengthen the evidence we presented to a great extent. However, before these models can be implemented in real vehicles, we need to be sure that that underlying vehicle controllers work safely and correctly. Also, we need to test and validate our models in terms of stability – when many vehicles on the road constantly change controllers, this may result in unstable behaviour.
This paper proves that it is in principle possible and beneficial to employ multi-objective evolution to develop useful sets of controllers as described; it does not claim that our NEAT-PS method is the best way to achieve this. We extended a well-known algorithm for the evolution of neural networks (NEAT) with multi-objective capabilities based on the Pareto-strength approach. We did not put any effort in tuning the algorithm settings to optimise performance. It is very likely that alternative EA implementations and/or precise tuning of the EA settings will result in controllers that deliver even better performance.
Part III

Multi-Objectivation
In this chapter, we generalise the method we used for multi-objectifying NEAT in chapter 5. Our method of multi-objectivation is independent of the scale or relative importance of the separate objectives. The basis of our method is the scalarisation of multiple objective values into a single number. We test two fitness assignment strategies, one based on Pareto strength and one on non-dominated sorting. We plug these fitness assignment strategies into the well-known CMA-ES algorithm and test the resulting algorithm on a suite of standard multi-objective test problems. On some test functions, this method yields good results, while on others, it gets stuck in local optima or covers only part of the optimal Pareto front. While this method proved to work well on the evolutionary algorithm that we described in chapter 5, our hypothesis that this method can be used as a general multi-objectifier was rejected.
Chapter 6. A Generic Method for Multi-Objectivation

6.1 Introduction

Recent advances in research into objective functions that promote (phenotypical) diversity in evolutionary computing such as novelty search (Lehman and Stanley, 2011) have (re-)kindled interest in multi-objectivation – adding support for optimising multiple objectives to algorithms that are in principle single objective.

Re-engineering an evolutionary algorithm’s selection strategies to match those of established multi-objective algorithms such as NSGA-II ( Deb et al., 2002 ) is not always a trivial task, particularly in sophisticated evolutionary algorithms that employ niching and/or speciation techniques. Also, a researcher may want to use an established implementation without having to delve into the intricacies of its selection schemes to rewrite them, with the attendant risk of introducing bugs.

An attractive solution to add multi-objective capabilities to an otherwise single-objective evolutionary algorithm would be to express an individual’s performance across objectives in a single agglomerated number that can then serve as the fitness value: such a method would not require detailed knowledge, let alone re-engineering, of the basic evolutionary algorithm’s selection mechanisms. Herein lies the attraction of the weighted-sum approach, but that approach requires a priori information about the possible values of the individual objective functions as well as their relative importance, along with other drawbacks ( Deb , 2001 ).

We investigate two possible methods to reduce multiple objectives to a single number and so provide a very straightforward, generally applicable method to allow evolutionary algorithms that were devised for single-objective scenarios to support multiple objectives. The first approach borrows from NSGA-II: it uses an individual’s rank after non-dominated sorting as the fitness value, the second method uses SPEA2’s Pareto Strength approach ( Zitzler et al., 2002 ). We apply both methods to an established, state-of-the-art numerical problem solver, CMA-ES ( Hansen et al., 2003 ). We then test the resulting algorithm on a number of multi-objective benchmark functions.

The purpose of our investigation is not to exceed or even match the performance of purpose-built multi-objective algorithms on these benchmark functions with our straightforward adaptation of CMA-ES. Rather, we want to establish whether simple selecting individuals on the basis of Pareto strength or their position after non-dominated sorting yields acceptable results and so offers a cheap fix-up for multi-objectivation.
6.2 Background

There are many well-established multi-objective evolutionary algorithms, for instance NSGA-II and SPEA2 (Deb, 2001; Zitzler et al., 2002). These are often the algorithm of choice in multi-objective settings, in particular for numerical optimisation problems.

There are, however, many types of problem where specific evolutionary algorithms that typically do not support multiple objectives are best suited and where the problem or the representation of the problem, is not directly applicable to one of the purpose-built multi-objective evolutionary algorithms that are available. For example, Mouret and Doncieux (2012) argued that there is much benefit to gain from adding novelty search as an extra objective to an evolutionary algorithm, thus effectively turning the problem in a multi-objective one. This multi-objectivation could benefit many more, otherwise single-objective problems. However, this multi-objectivation itself is not a trivial problem, since many evolutionary algorithms have very intricate mechanisms that do not necessarily allow for multi-objectivation in an easy way. The NEAT algorithm is an example of this.

Our interest in a simple fix-up to support multiple objectives stems from research into autonomous vehicle controllers as described in van Willigen et al. (2013c). There, evolution is employed to optimise neural net-based controllers to combine speed with comfort. To evolve neural nets (in this case to serve as controllers for autonomous driving), NEAT (Stanley and Miikkulainen, 2002) is currently one of the most popular and effective algorithms available and because it also evolves the network topology, there is no need to redefine the neural net controller topology. Available NEAT implementations, however, do not support multiple objectives and it is not straightforward to combine NEAT with, for instance, NSGA-II. NSGA-II includes an elaborate parent selection process, that takes care of maintaining diversity in the population. NEAT, on the other hand, has a completely different method of maintaining diversity in the population that is based on the actual neural structures of the individuals. It is impossible to combine these two approaches without compromising the very thing that defines the algorithms in their own right: the result would favour either NSGA-II (and not look at the internal structures of the neural networks) or NEAT (and not make use of the multi-objective diversity maintaining mechanisms that define NSGA-II).
Figure 6.1 – A graphic depiction of the proposed fix-up method. The evolutionary algorithm of choice (“Base EA”) creates and maintains the population. Individuals in the population are evaluated on all objectives and either the Pareto strength or the non-dominated sorting fitness value is assigned. The base EA then uses this assigned fitness to select survivors and parents on which to base a new population.

We investigate two methods for scalarisation of multiple objectives: non-dominated sorting and Pareto strength. Both methods compute a single scalar fitness value for each individual that summarises its performance on all the objective functions. Using this single number as the basis for selection in an arbitrary (single-objective) evolutionary algorithm is a straightforward and obvious proposition as illustrated in Figure 6.1.

The basis of both scalarisation methods is the well-known domination relation:

\[ i \succ j \iff \forall o \in O : o_i \succeq o_j \land i \neq j \]  \hspace{1cm} (6.1)

where the \( \succeq \) relation between objectives specifies that objective value \( o_i \) is equal or better than objective value \( o_j \), without saying whether the objective should be maximised or minimised.

### 6.2.1 Non-Dominated Sorting

Non-dominated sorting provides the basis for selection in the non-dominated sorting algorithm from NSGA-II (Deb et al., 2002). This algorithm assigns fitness value 1 to all non-dominated members in the population. It then assigns fitness value 2 to all members in the population that are only dominated by individuals with fitness 1 and so on until all individuals have been assigned a fitness value.
The assigned values reflect the rank of the front that each individual is on and this scalar fitness value indicates the relative performance of an individual across all objectives. Figure 6.2(a) illustrates this method.

### 6.2.2 Pareto Strength

The Strength Pareto Evolutionary Algorithm (SPEA-2) (Zitzler et al., 2002) scalarises an individual’s performance over multiple objectives by calculating the Pareto strength for each individual $i$ in population $P$ as follows:

$$S(i) = |\{ j | j \in P \land i \succ j \}|$$  \hspace{1cm} (6.2)

where $|\cdot|$ denotes set cardinality and $\succ$ denotes the dominance relation. The strength $S$ of an individual is the number of other individuals in the population that it dominates. Now we can calculate the fitness value of each individual $i$ by summing the strengths of the individuals $j$ that dominate $i$:

$$R(i) = \sum_{j \in P, j \succ i} S(j)$$  \hspace{1cm} (6.3)

Thus, if an individual is not dominated, its fitness value is 0. Figure 6.2(b) illustrates this method of assigning fitness values.

Pareto strength values are generally more informative than those assigned through non-dominated sorting. Not only do individuals get a fitness assignment based on how close they are to the non-dominated set of individuals, but the metric also includes information about the density of the Pareto front at different points.

### 6.3 Extending CMA-ES

In van Willigen et al. (2013c), multi-objectifying NEAT by simply plugging in the Pareto strength fitness evaluation strategy yielded good results. The resulting algorithm finds a dense Pareto front where each non-dominated individual corresponds with a controller that makes a unique distinction between the various objectives on the road. Now, we want to see whether or not this method also holds up when testing it on well-known multi-objective benchmark problems. This al-
Chapter 6. A Generic Method for Multi-Objectivation

(a) The non-dominated sorting strategy in a minimisation problem. An individual is assigned fitness \( n \) according to the \( n \)th non-dominated front the individual belongs to.

(b) The Pareto-strength strategy in a minimisation problem. First, we count the for each individual number of individuals that he dominates (the strength value, denoted by the black numbers left from the individuals). The fitness value for each individual is then the sum of the strength values of the individuals he is dominated by (denoted by the red numbers right from the individuals).

Figure 6.2 – The Pareto strength and the non-dominated sorting strategies.
allows for comparison with algorithms that provide native support for multiple objectives.

To allow for such a comparison, we turn to numerical optimisation problems. We apply the same plug-and-play method to CMA-ES (Hansen et al., 2003). This evolution strategy is a state-of-the-art algorithm that has proved particularly good at challenging numerical optimisation problems. Keeping the CMA-ES algorithm more or less intact, we plugged in the Pareto strength and non-dominated sorting multi-objective fitness assignment functions.

The JAVA implementation of the CMA-ES algorithm was downloaded from https://www.lri.fr/~hansen/cmaes_inmatlab.html#java and was amended in two places. The first change is trivial: the original CMA-ES code implements a predefined stop condition (finding the optimal solution) and if an individual gets within a certain margin of the optimal solution, the algorithm terminates. However, in multi-objective optimisation, the goal is not to find a single solution, but a range of Pareto-optimal solutions. Moreover, because the fitness values that this method assigns are relative to the rest of the population and thus there will be multiple optimal solutions in every single generation by definition, namely, the current non-dominated set of the population.

The second change to the original CMA-ES implementation is at the survivor selection stage of the algorithm. The problem here is more intricate: the original algorithm calculates the fitness for newly generated individuals and then replaces individuals in the original population with new individuals with a better quality. However, in our fitness evaluation scheme, the fitness of an individual is always relative to the entire population. This means that the quality of new individuals can only be compared if the dominance-based fitness values for the current generation are re-calculated to reflect domination in the union of new individuals and the current generation. Without this, there is no selection pressure towards better non-dominated individuals, because individuals in the non-dominated set of any population have exactly the same –optimal– fitness value.

6.4 Experiments and Results

We tackle the benchmark functions for multi-objective numerical optimisation listed in Deb et al. (2002), see Table 6.1. This allows us to provide a rough in-
dication the performance of the extended CMA-ES implementations compared to that of NSGA-II, which did well on all the problems in this test suite.

Table 6.1 – Test problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>n</th>
<th>variable bounds</th>
<th>Objective Functions</th>
<th>Optimal Solutions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCH1</td>
<td>1</td>
<td>[−10, 10]</td>
<td>$f_1(x) = x^2$, $f_2 = (x - 2)^2$</td>
<td>$x \in [0.2]$</td>
<td>convex</td>
</tr>
<tr>
<td>FON</td>
<td>3</td>
<td>[−4, 4]</td>
<td>$f_1(x) = \exp \left( -\sum_{i=1}^{n} (x_i - \frac{1}{\sqrt{n}})^2 \right)$; $f_2(x) = \exp \left( -\sum_{i=1}^{n} (x_i + \frac{1}{\sqrt{n}})^2 \right)$</td>
<td>$x_1 = x_2 = x_3$</td>
<td>non-convex</td>
</tr>
<tr>
<td>POL</td>
<td>2</td>
<td>[−π, π]</td>
<td>$f_1(x) = 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2$; $f_2(x) = |x_1 - 5|^2 + (x_2 + 1)^2$</td>
<td>Deb (2001)</td>
<td>non-convex, disconnected</td>
</tr>
<tr>
<td>KUR</td>
<td>3</td>
<td>[−5, 5]</td>
<td>$f_1(x) = \sum_{i=1}^{3} \left( 10 \exp \left( -0.2 \sqrt{x_i^2 + x_i^3} \right) \right)$; $f_2(x) = \sum_{i=1}^{3} \left( 1 + \sum_{i=1}^{n} \frac{1}{9} \right)$</td>
<td>Deb (2001)</td>
<td>non-convex</td>
</tr>
<tr>
<td>ZDT1</td>
<td>30</td>
<td>[0, 1]</td>
<td>$f_1(x) = x_1$, $f_2(x) = \exp \left( -\sum_{i=1}^{n} x_i \right) / (e - 1)$</td>
<td>$x_1 \in [0, 1]$</td>
<td>convex</td>
</tr>
<tr>
<td>ZDT2</td>
<td>30</td>
<td>[0, 1]</td>
<td>$f_1(x) = x_1$, $f_2(x) = \exp \left( -\sum_{i=1}^{n} x_i \right)$</td>
<td>$x_1 \in [0, 1]$</td>
<td>non-convex</td>
</tr>
<tr>
<td>ZDT3</td>
<td>30</td>
<td>[0, 1]</td>
<td>$f_1(x) = x_1$, $f_2(x) = \exp \left( -\sum_{i=1}^{n} x_i \right)$</td>
<td>$x_1 \in [0, 1]$</td>
<td>convex, disconnected</td>
</tr>
<tr>
<td>ZDT4</td>
<td>10</td>
<td>[0, 1]</td>
<td>$f_1(x) = x_1$, $f_2(x) = \exp \left( -\sum_{i=1}^{n} x_i \right)$</td>
<td>$x_1 \in [0, 1]$</td>
<td>non-convex</td>
</tr>
<tr>
<td>ZDT5</td>
<td>10</td>
<td>[−5, 5]</td>
<td>$f_1(x) = x_1$, $f_2(x) = \exp \left( -\sum_{i=1}^{n} x_i \right)$</td>
<td>$x_1 \in [0, 1]$</td>
<td>non-convex</td>
</tr>
<tr>
<td>ZDT6</td>
<td>10</td>
<td>[0, 1]</td>
<td>$f_1(x) = 1 - \exp \left( -4 \sin^2 \left( \frac{\pi x_1}{2} \right) \right)$; $f_2(x) = \exp \left( -\sum_{i=1}^{n} x_i \right)$</td>
<td>$x_1 \in [0, 1]$</td>
<td>non-convex, uniformly</td>
</tr>
</tbody>
</table>

We ran our constructed multi-objective version of CMA-ES with the two fitness assignment strategies as described above: Pareto strength and non-dominated sorting. We tested this algorithm on the 9 test functions listed in Table 6.1. We used default parameter settings for CMA-ES as listed in Table 6.2.

Figures 6.3 and 6.4 show the results of representative runs (out of 10 repeats) of the experiments. Each (horizontal) pair of graphs shows the results for one test function using CMA-ES combined with the Pareto strength (on the left) and the non-dominated sorting approach (on the right). In each graph, the orange line depicts the theoretical Pareto-optimal front and the blue dots depict the objective values of the complete population at the 1000th generation.

We refrain from detailed statistical analyses of the results because in many cases the difference between the theoretical optima as well as the results reported for NSGA-II (Deb et al., 2002) are so large that a qualitative analysis seems appropriate.
Figure 6.3 – The performance of extended CMA-ES on several multi-objective functions. The orange line is the actual Pareto-optimal front and the blue dots are the performance of the population of a representative run after 1000 generations.
Figure 6.4 – The performance of extended CMA-ES on several multi-objective functions. The orange line is the actual Pareto-optimal front and the blue dots are the performance of the population of a representative run after 1000 generations.
Table 6.2 – Experimental Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>100</td>
</tr>
<tr>
<td>Number of generations</td>
<td>1000</td>
</tr>
<tr>
<td>Test problems</td>
<td>See Table 6.1</td>
</tr>
<tr>
<td>Fitness assignment</td>
<td>Pareto Strength and Non-dominated Sorting</td>
</tr>
<tr>
<td>Initial $x_i$ values</td>
<td>0.5</td>
</tr>
<tr>
<td>Initial $\sigma_i$</td>
<td>0.05</td>
</tr>
<tr>
<td>Upper $\sigma_i$ boundary</td>
<td>0.5</td>
</tr>
<tr>
<td>Lower $\sigma_i$ boundary</td>
<td>0.001</td>
</tr>
<tr>
<td>Number of repeats</td>
<td>10</td>
</tr>
</tbody>
</table>

For the first test function, ZDT1, the whole of the population lies fairly close to the Pareto-optimal front. Pareto strength seems to slightly outperform non-dominated sorting, but the difference is not substantial. In ZDT2, both populations converge very rapidly to a single point on the Pareto front, optimising only the first objective. ZDT3 has a disconnected Pareto front and on all runs for both strategies, the population focusses on a single segment of the disconnected Pareto front. ZDT4 is a hard problem that has many local optima, each with an attractive Pareto front. Both strategies converge on a single local front that they cannot escape from. The population in ZDT6 converges very rapidly to the optimal solution in one objective, but after that the algorithm is not able to also optimise on the second objective. On both SCH and FON, the algorithm converge on the Pareto front with a dense non-dominated population. POL and KUR are both disconnected test functions, which our method has trouble with, just as with ZDT3. In POL, a local sub-optimal front is found and in KUR, similarly to ZDT3, the algorithm converges on a single segment of the disconnected Pareto front.

6.5 Discussion

The results vary greatly per test problem. On some test problems, such as ZDT1, SCH and FON, the algorithm yields a population that lies on or near the optimal Pareto front. On other problems, however, the algorithm performs very badly, getting stuck in local minima or converging on a small portion of the optimal Pareto front.
Strikingly, there is little or no difference between the results using Pareto strength and those using non-dominated sorting, even though, as mentioned earlier, Pareto strength could be expected to yield somewhat better results because it provides more information about the spread of solutions near the Pareto front. We hypothesise that this is a trait of CMA-ES. The algorithm typically finds a very good spread of solutions across (parts of) the Pareto front. This implies that only a fraction of the population consists of dominated individuals. When this is the case, there is hardly any difference in the fitness assignment of Pareto strength versus non-dominated sorting: in both cases, the non-dominated set has optimal fitness.

We see some trends in the performance on different test functions. First of all, if the Pareto front is convex (ZDT₁ and SCH), the algorithm sometimes works well. On most problems with non-convex and on all problems with a disconnected Pareto front, however, performance drops dramatically. For example, on ZDT₂, the population converges in its entirety to one single point. On ZDT₃ and KUR, some parts of the Pareto front are found, but the algorithm never manages to discover multiple sections of the Pareto-optimal front. ZDT₄ is a deceptive function, with many local Pareto fronts and the algorithm is not able to get out of those. On ZDT₆, the population optimises rapidly on one objective, but is after that unable to optimise the second objective.

From the results that we found, we can conclude that this method is not a generally applicable, one-size-fits-all solution to multi-objectivisation of otherwise single-objective evolutionary algorithms. However, it retains some attraction because it is very simple to implement, making it primarily valuable for practitioners that wish to exploit available evolutionary algorithms without having to delve deeply into the inner workings of these algorithms.

In situations where it is easy to ascertain whether the resulting non-dominated solutions yield good or acceptable performance, this method provides a quick method of adapting an existing algorithm to see if a fix-up is at hand before investing in a much more in-depth revision of the algorithm (which is what Igel did for a ‘proper’ multi-objective implementation of CMA-ES (Igel et al., 2007)).

A solution to the problem of the algorithm finding only a very specific region of the Pareto-optimal front (e.g. ZDT₂), could be to add novelty search as a third objective to this algorithm. This could improve the spread of the found Pareto front. In the case of disconnected functions, this could find different regions of the
Pareto front and in the case of functions with very attractive local optima, it could help the algorithm to jump out of these local optima.

We did not tune our algorithm extensively and by large used default values of CMA-ES. We did this specifically to test if our method could be a plug-and-play solution to multi-objectification of evolutionary algorithms. We conclude that this is generally not the case, but tuning might be the answer to our problems. We do not know whether longer runs, larger populations or higher mutation rates will lead to better performance. Chances are that the performance of our method is increased when applying tuning methods. This is planned as future work.

6.6 Conclusion

This section explores a simple and generic method of multi-objectifying evolutionary algorithms. Endowing an evolutionary algorithm with the ability to deal with multiple objectives is not always a trivial task, particularly for algorithms that incorporate sophisticated selection techniques.

An attractive solution to this problem is to scalarise the fitness values of multiple objectives, since this only impacts the fitness assignment process. A classical way to achieve this is using the weighted sum of the separate objective values. The downside of this method is that it requires a priori information about the possible values of the objectives as well as their relative importance.

We investigated two different methods of scalarising fitness values that would offer a very straightforward way of making otherwise single-objective evolutionary algorithms support multiple objectives. The first approach is based on non-dominated sorting, borrowed from NSGA-II, the other exploits the Pareto strength algorithm from SPEA2. We applied both of these methods to CMA-ES, a very well-known, state-of-the-art numerical optimiser. We tested the algorithm on a test-suite of multi-objective functions.

The purpose of this work was to investigate if our quick fix-up for multi-objectification, that led to satisfying results in earlier work van Willigen et al. (2013c), would yield acceptable results on a suite of well-known numerical multi-objective problems.

Both scalarising approaches seem to work on a limited selection of test problems. If the solution space is convex, the algorithm works well. On the other hand, when the Pareto front is non-convex and/or disconnected, the algorithm generally
gets stuck in local optima or converge on good solutions for only one of the objectives. When the Pareto front is disconnected, only a single section of the Pareto front is covered by the population and different runs of the same algorithm may find a different segment of the Pareto front.

In terms of performance, there is almost no difference between the Pareto strength and the non-dominated sorting approaches. In a few cases, the Pareto strength method seems to outperform non-dominating sorting, but the difference is never substantial.

For future work, we plan to do some extensive parameter tuning on this method. It may improve the performance of our algorithm significantly, since we primarily used default values of the CMA-ES algorithm in our experiments.

Another branch of future work is to investigate whether novelty search as an extra objective could improve the performance of the algorithm. This could solve the problem of some test functions, where the whole population rapidly converges towards a single point on the Pareto front.

In many of the test functions, the algorithm converges rapidly towards one of the objectives and is afterwards unable to break out of that local optimum. We hypothesise that this may be a general problem with CMA-ES, because this algorithm reduces step sizes as the population converges to a single point. When this happens, step sizes may simply be too small for the population to explore the rest of the Pareto front. We intend to investigate this matter further.
7

Summary and Conclusions

7.1 Look Ma, No Hands!

In this thesis, we explored various aspects of autonomous vehicle control. We started off in the domain of path planning in unmanned aerial vehicles, but the larger part of this thesis deals with autonomous cars on highways.

In this chapter, we re-iterate and assess the objectives as stated in the introduction and place the scientific contributions of this research in a broader context.

1. Identify applications within the transportation systems domain where autonomy can be beneficial. We have identified three applications within the transportation systems domain in which varying levels of autonomy are beneficial. Without autonomy, it is substantially harder for humans to make informed decisions, especially when optimisation on multiple objectives is desired. The first application is path planning for unmanned aerial vehicles (UAVs). The second application is safety checking in platooning on highways, under uncertain conditions. Uncertainty in information, communication and system behaviour makes it hard to estimate the critical headway time that guarantees safety in real time. The third application is high-level
autonomous vehicle control on highways that meets the changing needs of drivers.

2. **Develop algorithms that are useful in these applications and develop demonstrators to test the algorithms.** In all aforementioned applications, we developed algorithms that add some level of autonomy. We developed simulators for these applications, that serve as demonstrators.

3. **Implement, test and assess the quality of these algorithms and, if possible, compare them to existing benchmark algorithms.** Using the demonstrators, we were able to assess the quality of the algorithms. In most applications, no benchmark algorithms were available to compare our own work with. However, in these cases, we modelled the current practice in these domains, which could then serve as benchmark for our algorithms.

Per application, we shortly describe what our work consists of and what the main conclusions are.

### 7.1.1 Autonomous UAVs

We developed an algorithm for autonomous path planning for UAVs. This algorithm uses a particle swarm optimiser to determine a good flight path before the UAV starts flying. Then, during the flight, this pre-determined flight path may change based on observations during the flight.

The algorithm that we used to optimise these flight paths had multiple goals: on the one hand, the number of observed targets had to be optimised, but on the other hand, targets also had to be observed as well as possible. The practical implementation of this problem was a weighted function that specified the relative importance of those two goals.

### 7.1.2 Safety in the Face of Uncertainty

Within the domain of intelligent highway systems, we worked on a model that computes the critical headway time between two platooning vehicles in real time. This analytical model can take all kinds of uncertainties in information, communication and system behaviour into account. We extensively tested the model in simulation and compared the results with a rigorous Monte Carlo method. We concluded that the analytical, real-time method works just as well.


7.2 Contributions of the Thesis

This thesis contains the following scientific contributions. In the domain of path planning for autonomous UAVs, we constructed an algorithm that a) determines a priori the best flight given some objective function and some prior knowledge of the terrain and b) when the UAV is in the air, it can slightly adjust its flight path based on actual observations. This algorithm is useful in situations where unknown terrains have to be searched for objects. Beforehand, some assumptions can be made about some terrain, based on which an a-priori search path can be determined. However, during the flight, the situation could be different from what you expected beforehand. This algorithm can change the pre-defined search path if necessary. In situations where it is expensive or dangerous to send humans in the field, this is a good alternative.

Another contribution is a method that calculates the critical safe headway time between two vehicles, under uncertain conditions, in real time. This method is useful in the domain of cooperative adaptive cruise control, where vehicles on highways can use direct communication and other sensors to be able to drive closely to each other. The method described in this thesis can be used while vehicles are platooning, i.e., following other vehicles autonomously, using short distances. To ensure safety in this application, being able to deal with uncertainty in information, communication and system behaviour is crucial.

The next scientific contribution described in this thesis also lies within the domain of autonomous vehicle control on highways, albeit on a higher level of control. We provided an algorithm that evolves sets of controllers for autonomous vehicles. The novelty of the algorithm lies in that it creates controllers that take driver
preferences into account. Different drivers have different objectives: some prefer going fast, others prefer comfort or fuel economy. We constructed an algorithm that evolves multiple controllers, each signifying a unique prioritisation between these preferences. Vehicles can then select the appropriate controller that adheres best to the preferences of the driver. This work shows that (high-level) controller design for autonomous vehicles can be done automatically. Giving drivers the freedom to customise their vehicle’s behaviour increases user acceptance of such systems.

The final contribution described in this thesis is extensive experimentation with a generalised version of the earlier mentioned multi-objective evolutionary algorithm. We tested if we could multi-objectify a different evolutionary algorithm by using the same method as in chapter 5. The results of these experiments did not verify the hypothesis that our method of multi-objectivation works in various settings. Still, we think it is a valuable contribution to the domain of multi-objective evolutionary computing.

### 7.3 Future Research

The research posed in this thesis opens up some interesting directions for future research. Every algorithm and every experiment for this thesis was performed in simulation. One of the most obvious paths of future research is to implement these algorithms in real vehicles and see if the algorithms still hold up. For the algorithm that calculates safety under uncertain circumstances, this could be done relatively easily. The technology and testbeds already exist to implement this feature and only two vehicles are needed to do initial testing. Testing the preference-based controller design algorithm is much harder. Not only a lot of vehicles are needed, each equipped with platooning and autonomous driving capabilities, but also a long stretch of road and available testers.

Another big branch of future research would be to see if the preference-based controllers could be evolved while being on the road. In this thesis, we identified a couple of possible objective functions for drivers and evolved the controllers in an off-line setting: populations of controllers were tested in simulation and after some generations, a set of controllers was found. It would be interesting to see if these controllers could be evolved while driving. Learning on the fly has some great advantages: vehicles could adapt to changing situations on the road. In differ-
ent countries, people have different driving behaviours and on-the-fly adaptation could find controllers that differ per country. One may think that safety would also be a big challenge when controllers evolve on the road. However, these high-level vehicle controllers could never become dangerous for the driver, because safety-critical procedures such as obstacle avoidance are organised at the lower levels of control. This is evident in chapter 5, where actions of the neural network simply are not executed when they are not possible or dangerous.

In addition, extensive tuning of the algorithms that we describe in this thesis is another direction for future research. Tuning algorithms is proven to be highly useful (Smit, 2012). When tuning an algorithm, better sets of parameters could be found, generally yielding a better performance of the algorithm. Having tuned algorithms gives much better insight in how algorithms may compare to other algorithms. Some untuned algorithm $a$ may seem worse at first sight than algorithm $b$, while the tuned version may turn out to be much better.

As a final note, we like to re-iterate our vision of future transportation systems. Autonomous vehicles, especially on the road, are an important part of our future. This thesis addresses some important aspects of getting closer to this future. Some contributions in this thesis are relevant in the development of current systems, while other contributions become relevant in a more distant future. But we do not doubt that this future is upon us.


UNECE (2013). Intelligent transport systems (its) for sustainable mobility.


References


Kijk mam, zonder handen!

Aspecten van Autonomie in Voertuigen

Te land, ter zee en in de lucht: we zijn omringd door transportatiesystemen. Een imiserende wereldbevolking en verregaande globalisering zorgt ervoor dat men steeds meer afhankelijk wordt van efficiënte transportatiemethoden.

Dit proefschrift gaat over autonome voertuigen, waarbij er geen mens bij de besturing aan te pas komt. Enkele relevante vragen die hierbij komen kijken: Hoe plan je de route van een autonoom vliegtuigje? Hoe zorg je ervoor dat een zelfrijdende auto op tijd remt? Hoe zorg je ervoor dat een zelfrijdende auto zich houdt aan de rijstijl die de bestuurder de auto oplegt? Dit zijn enkele vragen waar dit proefschrift over gaat.

Het proefschrift bestaat uit drie delen. Het eerste deel gaat over autonome, onbemande vliegtuigjes. Als er een afweging gemaakt moet worden tussen exploitatie (bekijk zo veel mogelijk) en exploitatie (bekijk zo goed mogelijk) van een onbekend terrein, is het van belang dat het vliegtuig tijdens de vlucht een keuze kan maken om op basis van observaties (bijvoorbeeld: een interessant object wordt gesignaleerd) in de buurt te blijven, in plaats van de oorspronkelijke route te blijven vliegen. Voor dit stuk onderzoek heb ik gebruik gemaakt van een Particle Swarm Optimiser (PSO) die, voordat de vlucht daadwerkelijk plaatsvindt, een optimale route bepaalt op basis van bepaalde voorkennis over het te verkennen terrein. Op het moment dat het vliegtuigje in de lucht is, treedt een mechanisme in werking dat een afweging maakt tussen doorvliegen of rondcirkelen boven een object, op het moment dat dat gedetecteerd wordt.

Het tweede en grootste gedeelte van mijn proefschrift gaat over zelfrijdende auto’s op snelwegen. Een trend die al langere tijd aan de gang is, is de transitie naar intelligente transportatiesystemen, waarin auto’s steeds autonome te werk gaan. Radars, Wi-Fi, GPS en andere sensoren kunnen gegevens over locatie en snel-
heid van de omringende auto’s meten en zo de zelfrijdende auto informeren over een aan te houden afstand. Een belangrijk aspect hiervan is veiligheid. Immers, sensoren zijn niet perfect: radars kunnen meetfouten maken, Wi-Fi kan uitvallen. Dit proefschrift beschrijft een methode waarmee je, gegeven deze onzekerheid van je sensoren, kan berekenen wat de minimale afstand is die je tot je voorganger moet aanhouden.

Een ietwat futuristischer hoofdstuk van dit proefschrift gaat over behoeftes van bestuurders die vertaald kunnen worden naar rijstijlen van zelfrijdende auto’s. Sommige bestuurders willen hard rijden, anderen juist comfortabel, en voor weer anderen is zuinigheid een belangrijke doelstelling. Deze doelstellingen zijn vaak conflicterend. Ik beschrijf een methode waarmee er meerdere regelaars voor zelfrijdende auto’s worden gemaakt, die elk een unieke prioritering van doelstellingen vertegenwoordigt.

In het laatste deel van mijn proefschrift behandel ik een generalisatie van de methode die de meerdere doelstellingen van bestuurders probeert te optimaliseren. Deze multi-objective evolutionaire methode heb ik losgelaten op een verzameling standaardproblemen.

Autonome voertuigen worden een steeds belangrijker onderdeel van onze toekomst. Dit proefschrift behandelt een aantal belangrijke aspecten om dichter bij deze toekomst te komen. Sommige bijdragen van dit proefschrift zijn relevant voor bestaande systemen, en andere bijdragen worden pas relevant in een wat verdere toekomst. Maar we twijfelen er niet aan dat deze toekomstvisie werkelijkheid wordt.
SIKS Dissertatiereken

2009
- 01 Rasa Jurgelevičaitė (RUN), Symmetric Causal Independence Models
- 02 Willem Robert van Hage (VU), Evaluating Ontology-Alignment Techniques
- 03 Marc Schoonderwoerd (VU), A Framework for Evidence-Based Policy Making Using Collaboration Engineering
- 04 Josephine Nubukonya (RUN), Improving the Quality of Organizational Policy Making using Collaboration Engineering
- 05 Jelte Overboek (RUN), Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality
- 06 Maksym Korotkiy (VU), Understanding Classification
- 07 Ronald Poppe (UT), Discriminative Vision-Based Recovery and Recognition of Human Motion
- 08 Hans Bloem (UVa), Evolutionary Agent-Based Policy Analysis in Dynamic Environments
- 09 Benjamin Kanavag (RUN), Design, Discovery and Construction of Service-oriented Systems
- 10 Jan Wijkstra (VU), Logic programming for knowledge-intensive interactive applications
- 11 Alexander Boer (VU), Legal Theory. Sources of Law & the Semantic Web
- 12 Peter Massuth (TUE, Humboldt-Universität zu Berlin), Operating Guidelines for Services
- 13 Steven de Jong (UvA), Fairness in Multi-Agent Systems
- 14 Mak ym Korotkiy (VU), From onto-logy-enabled services to service-able ontologies (making ontologies work on e-science with ONTO-SOA)
- 15 Rinke Huijzer (VU), Ontology Representation - Design Patterns and Ontologies that Make Sense
- 16 Josephine Nabukonya (RUN), Improving the Quality of Organizational Policy Making using Collaboration Engineering
- 17 Yves Thonnart (UT), Network Architectures in Computer Chess
- 18 Laurens van der Maaten (UvT), Feature Extraction from Visual Data
- 19 Fabian Groth (CWI), Armadillo, An Everliving Database System
- 20 Valentin Robus (CWI), Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets
- 21 Rob van der Vecht (UvT), Adjustable Autonomy: Controlling Influences on Decision Making
- 22 Ingrid Vorenhout (LUM), Ranking and Reliable Classification
- 23 Pavol Sendyukov (UT), Search For Expertise: Going through direct evidence
- 24 Peter Holsopple (VU), Modelling Web-Usage in a Changing Environment
- 25 Anwar Nasser (VU), Cognitive Models for Training Simulations
- 27 Fred Dewulf (Essex U), An Agent-Based Model for the Development of Intelligent Mobile Services
- 28 Christian Gehr (U), Contextual Support of social Engagement and Reflection on the Web
- 29 Sander Evers (UT), User Data Management with Probabilistic Models
- 30 Stanislav Pokraev (UT), Model-Driven Semantic Integration of Service-Oriented Applications
- 31 Moritz Zuckowski (CWI), Balancing vectorized query execution with bandwidth-optimized storage
- 32 Sohia Katsir (UvA), A Closely Look at Learning Relations from Text
- 33 Rik Farenhorst (VU) and Remco de Boer (VU), Architectural Knowledge Management: Supporting Architects and Auditors
- 34 Khiel Truong (UT), How Do Real Affect Affect Affect Recognition In Speech?
- 35 Menno van de Wouw (UvA), Advances in Software Product Management: An Incremental Method Engineering Approach
- 36 Wouter Koolen (UT), Privacy on Political/Internet Forums: Do you know what a search engine is doing?
- 37 Marco Kalz (RUN), Placement Support for Learners in Learning Networks
- 38 Jochen Brack (RUN), Navigation Support for Learners in Informal Learning Networks
- 39 Rina Vercruijsse (UvU), ‘Tags and self-organisation: a metadata ecology for learning resources in a multilingual context
- 40 Hugo Kielman (UL), Service Substitution – A Behavioural Approach Based on Petri Nets
- 41 Steph an Balken (VU), Multimodal Language Learning: Investigations into the Geometry of Language
- 42 Igor Berezhnoy (UT), Digital Analysis of Paintings
- 43 Joanne Roos (UT), Recommender Systems for Social Bookmarks
- 44 Virginia Nunes Leal Franzquiera (UT), Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Agents
- 45 Roberto Santana Tapia (UT), Assessing Business-IT Alignment in Networked Organizations
- 46 Loredana Almajan (UvA), Querying XML: Benchmarks and Recursion

2010
- 01 Matthijs van Leussen (UvU), Patterns that Matter
- 02 Ingo Wissink (UvT), Work flows in Life Science
- 03 Joost Geurts (CWI), A Document Engineering Model and Processing Framework for Multimedia documents
- 04 Nils Olof Källvik (UvA), Knowledge is Power: Situational Awareness of Co-located Teams in Multisite Environments
- 05 Claudia Hoff (UT), Predicting the Effectiveness of Queries and Retrieval Systems
- 06 Sander Bakker (UT), Rapid Adaptation of Video Game AI
- 07 Wim Fikkers (UT), Gesture interaction at a Distance
- 08 Krzysztof Siecieczka (UvU), Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and governments
- 09 Hugo Kielman (UL), A Political governance-on-the-go, Privacy and e-traffic surfering
- 10 Rebecca Ong (UL), Mobile Communication and Protection of Children
- 11 Adriaan Ter Mors (TUD), The world according to MARP. Multi-Agent Route Planning
- 12 Susan van den Brink (U), Sensemaking software for crime analysis
- 13 Gianluigi Faloni (RUN), High Performance Data Mining using Bio-inspired techniques
- 14 Sander van Splinter (VU), Service-based Service Recomposition
- 15 Lieke Bodenhof (UT), Managing Dependency Relations in Inter-Organizational Models
- 16 Nico Verver (TUD), Efficient Identification of Timed Automata, theory and practice
- 17 Spyros Kotsouros (VU), Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications
- 18 Charlotte Gerritsen (VU), Caught in the Act: Investigating Crime by Agent-Based Simulation
- 19 Henriëtte Cramer (UvA), People’s Responses to Anonymous and Adaptive Systems
- 20 Ivo Swartjes (UT), Whose Story Is It Anyway? How Improve Information Agency and Authorship of Emergent Narrative
- 21 Harald van Hoeken (UT), Privacy-aware data management by means of data degradation
- 22 Michael Hildebrand (CWI), End-user Support for Access to Heterogenous Linked Data
- 23 Bas Steunebrink (Uu), The Logical Structure of Emotions
- 24 Dimitry Tykdenaar, Designing Generic and Efficient Negotiation Strategies
- 25 Fadiqar Ali Memon (UvU), Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective
- 26 Ying Zhu (CWI), XRPC: Efficient Distributed Query Processing on Heterogenous XML Engines
- 27 Marion Voulon (Uu), Automatically contracten
- 28 Arne Koopman (Uu), Characteristic Relational Patterns
- 29 Sotiris Iliadis (CWI), Database Cracking: Towards Auto-tuning Database Kernels
- 30 Marno van Erp (UvT), Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval
- 31 Victor de Boer (UvA), Ontology Enrichment from Heterogenous Sources on the Web
- 32 Marcel Heijl (UT), An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems
- 33 Robin Ahl (UT), Modelling Representation Uncertainty in Concept-Based Multimedia Retrieval
- 34 Teddie Dingbáty (UT), Interaction Design in Service Compositions
- 35 Dolf Tresch (U), Proof of Concept: Concept-based Biomedical Information Retrieval
- 36 Jose Jansen (UvU), Paving the Way for Lifelong Learning: Facilitating competence development through a learning path specification
- 37 Niels Lohmann (TUE), Correctness of services and their composition
2012-38 Erik Fahland (TUE), From Simcario to components
2012-39 Ghaitha Farooq Siddiqui (VU), Interoperability modeling of emotions in virtual agents
2012-40 Mark van Assem (VU), Converting and Integrating Vocabularies for the Semantic Web
2012-41 Guillaume Chachet (UM), Menu-Card Tree Search
2012-42 Sybren de Kinderen (VU), Needs-driven service handling in a multi-supplier setting, the computational cs-service approach
2012-43 Peter van Kranenburg (TU), A Computational Approach to Content-Based Retrieval of Folk Song Melodies
2012-44 Pieter Belfskeurs (TUE), An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Televisie Domain
2012-45 Vasilios Andrikopoulos (UvT), A theory and model for the evolution of software services
2012-46 Vincent Pilgors (TU), Exploration: Inter-Organizational Business-ICT Alignment
2012-47 Chen Li (UT), Mining Process Model Variants: Challenges, Techniques, Examples
2012-48 Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets
2012-49 Johan Takooshi Saito (UM), Solving difficult game positions
2012-50 Basie Huzman (VU), Search in Audiovisual Broadcast Archives
2012-51 Ali Khaltiar Amin (CWI), Understanding and supporting information seeking tasks in multiple sources
2012-52 Peter-Paul van Maaren (VU), Adaptive Support for Human-Computer Teams: The Use of Cognitive Models of Trust and Attention
2012-53 Edgar Meij (UvA), Combining Concepts and Language Models for Information Access

2011-01 Botond Cscke (RUN), Vorational Algorithms for Bayesian Inference in Latent Gaussian Models
2011-02 Nick Thiemensmeier(UU), Organizing Agent Organizations: Syntas and Operational Semantics of an Organization-Oriented Programming Language
2011-03 Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems
2011-04 Hadi van Hasselt (CU), Insights in Reinforcement Learning: Formal analysis and empirical evaluation of temporal-difference learning algorithms
2011-05 Vicent Raad (TU), Enterprise Architecture Coming of Age - Measuring the Performance of an Emerging Discipline
2011-06 Yonin Wang (TUE), Semantically-Enhanced Recommendations in Cultural Heritage
2011-07 Xujia Cao (UT), Multi-modal Information Presentation for High Load Human Computer Interaction
2011-08 Nielske Vergunst (UvA), BDI-based Generation of Robust Task-Oriented Dialogues
2011-09 Tim de Jong (OC), Contextualized Mobile Media for Learning
2011-10 Bart Bogart (VU), Cloud Content Composition
2011-11 Dhuval Vyas (UT), Designing for Awareness: An Experience-focused HCI Perspective
2011-12 Carmen Beatozim (TUE), Grid Architecture for Distributed Process Mining
2011-13 Xiaoyu Mao (UvT), Airport under Control. Multilevel Scheduling for Airport Ground Handling
2011-14 Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets
2011-15 Martin Koolen (UvA), The Meaning of Structure: the Values of Link Evidence for Information Retrieval
2011-16 Maarten Schaud (UM), Selective Search in Games of Different Complexity
2011-17 Ihsan He (UvA), Exploring Topic Structure: Coherence, Diversity and Relatedness
2011-18 Mark Ponsen (CN), Strategic Decision-Making in complex games
2011-19 Ellen Rusman (CU), The Mind + Eye on Personal Profiles
2011-20 Dong Cai (VU), Guiding service-oriented software engineering. - A tree-based approach
2011-21 Linda Terlouw (TUD), Modularization and Specifiation of Service-Oriented Systems
2011-22 Jurrie Zhang (UvA), System Evaluation of Archival Description and Access
2011-23 Wouter Wiersma (VU), Finding People and their Utterances in Social Media
2011-24 Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Interaction
2011-25 Syed Waqar ul Quasam Jafri (AY), Analysis and Validation of Models for Trust Dynamics
2011-26 Matthias Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Investment-Distance Trade-Offs in Embodied Conversational Agents and Robots
2011-27 Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns
2011-28 Rianne Kaptopra(UVA), Effective Focused Retrieval by Executing Query Context and Document Structure
2011-29 Faisal Kaminat (TU), Discrimination-aware Classification
2011-30 Egon van den Broek (UT), Affective Signal Processing (ASP): Unmasking the mystery of emotions
2011-31 Lucio Cappell (EUR), Computational and Game-Theoretic Approaches for Modeling Bound Rationality
2011-32 Noos-Jan van Eck (EUR), Knowledge in the Web of Information: An Entity Based Approach
2011-33 Tom van der Woode (UU), Arguing to Motivates decisions
2011-34 Paolo Tarumi (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations
2011-35 Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training
2011-36 Erik van der Spek (UvT), Experiments in serious game design: a cognitive approach
2011-37 Adriana Burtuli (RUN), Machine Learning for Paractice Data, Applications for Preference Learning and Supervised Network Inference
2011-38 Nyos Lemmens (UM), Bio-inspired Distributed Optimization
2011-39 Jan Weistra (UU), Organizing Adaptation using Agents in Serious Games
2011-40 Viztoros Clerc (VU), Architectural Knowledge Management in Global Software Development
2011-41 Luau Brems (UT), Cryptographically Enforced Distributed Data Access Control
2011-42 Michal Sindlar (UU), Explaining Behavior through Mental Attribution
2011-43 Hiksr van de Schuur (CU), Process Improvement through Software Operation Knowledge
2011-44 Boris Readerink (UT), Robust Brain-Computer Interfaces
2011-45 Herman Stehouwer (UU), Computational Language Models for Alternative Sequence Selection
2011-46 Bebi Hs (TUD), Towards Contextualized Information Delivery: A Role-based Architecture for the Domain of Mobile Police Work
2011-47 Aziz Bin AbUzayd (VU), Exploring Computational Models for Intelligent Support of Persons with Depression
2011-48 Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
2011-49 Andrea Nicoleas (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012-01 Terry Kakeets (UvT), Relationship Marketing for SMEs in Uganda
2012-02 Muhammad Umar(UV), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
2012-03 Param Waranya (VU), Supporting Architecture Evolution by Mining Software Repositories
2012-04 Jurriaan Souer (UvT), Development of Content Management System using Web Applications
2012-05 Marvin Plopp (UT), Maturing Informational Information systems
2012-06 Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks
2012-07 Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
2012-08 Gerhard van Vries (UvA), Kernel Methods for Visual Trajectories
2012-09 Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms
2012-10 David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment
2012-11 I.C. Ramachandran (TUE) Process Mining in the Large: Progress, Discovery, and Diagnostics
2012-12 Kees van der Suijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems
2012-13 Saleemah Shahid (UT), Face and Face: Exploring non-verbal expressions of emotion during playful interactions
2012-14 Evgeny Knutov(TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems
2012-15 Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitve and Affective Processes
2012-16 Fumio Nakaya (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
2012-17 Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
2012-18 Elho Posser (VU), Improving Solution Architecting Practices
2012-19 Helen Schoenborn (TUE), What’s Next? Operational Support for Business Process Execution
Coordinating autonomous planning and scheduling
The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
Reasoning with Contexts in Description Logic
Ensemble Approaches to Semi-Supervised Learning Learning
Multiagent Learning - Dynamic Games and Applications
Exploring the practical benefits of argumentation in multi-agent deliberation
News Analytics for Financial Decision Support
Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
Handling Abrupt Changes in Evolving Time-series Data
Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
Fast and Reliable Online Learning to Rank for Information Retrieval
Quantifying Individual Player Differences
Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
Outlier Selection and One-Class Classification
Resource-Aware Business Process Management: Analysis and Support
Game Design Patterns for Learning
Risk-aware design of value and coordination networks
Architectural Support for Dynamic Homecare Service Provisioning
Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System
Minimal Mobile Human Computer Interaction
Pattern Mining in Data Streams
Ambient Learning Displays
Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
Exploration and Contextualization through Interaction and Concepts
Studies in Learning Monotone Models from Data
Coalitions in Cooperation Networks (COCOON)
Evolutionary analysis in PPI networks and applications
Combining System Dynamics with a Domain Modeling Method
Aligning Observed and Modeled Behavior
Information Retrieval for Children: Search Behavior and Solutions
Service Value Networks
Peer-to-Peer Information Retrieval
Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
2014-12 Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control
2014-13 Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains
2014-14 Yangyang Shi (TUD), Language Models With Meta-Information
2014-15 Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare
2014-16 Krystyna Milan (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria
2014-17 Kathrin Dentler (VU), Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability
2014-18 Mattijs Ghijsen (VU), Methods and Models for the Design and Study of Dynamic Agent Organizations
2014-19 Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support
2014-20 Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link
2014-21 Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments