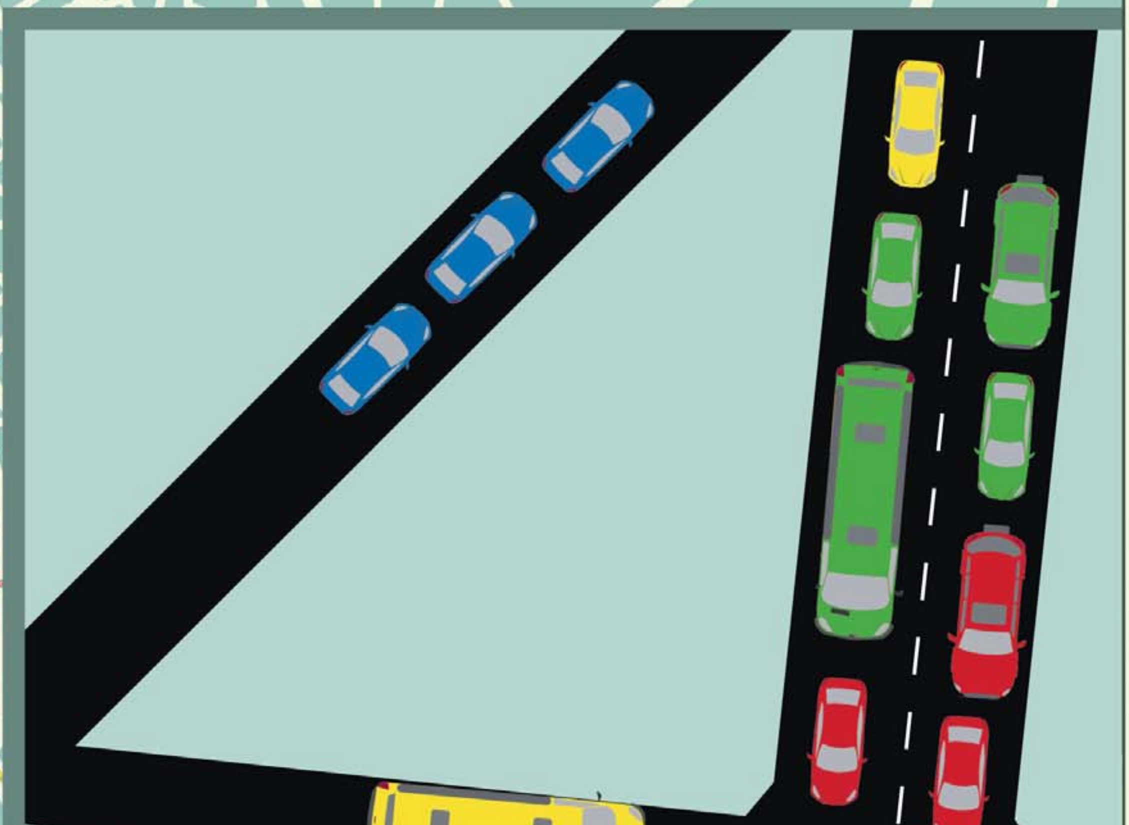
Jana Görmer-Redding

Autonomous Vehicle Groups in Urban Traffic



Cuvillier Verlag Göttingen
Internationaler wissenschaftlicher Fachverlag

Für meine Töchter

TU Clausthal
Clausthal University of Technology

Autonomous Vehicle Groups in Urban Traffic

DISSERTATION

zur Erlangung des Grades eines Doktors der Naturwissenschaften

vorgelegt von

Jana Görmer-Redding

geboren in Berlin, Deutschland

genehmigt von der Fakultät für Mathematik / Informatik und Maschinenbau
der Technischen Universität Clausthal

Vorsitzender der Promotionskommission: Prof. Dr. Jürgen Dix
Hauptberichterstatter: Prof. Dr. Jörg P. Müller
Berichterstatter: Prof. Dr. Sven Hartmann

Tag der mündlichen Prüfung am 07. Februar 2018

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

1. Aufl. - Göttingen: Cuvillier, 2018
Zugl.: (TU) Clausthal, Univ., Diss., 2018

© CUVILLIER VERLAG, Göttingen 2018
Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

1. Auflage, 2018
Gedruckt auf umweltfreundlichem, säurefreiem Papier aus nachhaltiger Forstwirtschaft

ISBN 978-3-7369-9828-5
eISBN 978-3-7369-8828-6

Summary

It is likely that autonomous vehicles will be the future of mobility. To handle the increase in autonomy, traffic coordination methods will become indispensable. Based on this, an investigation into the performance of Autonomous Vehicle Group Formation (AVGF) based on a decentralized model and a simulative evaluation in urban environments is needed. An Autonomous Vehicle Group (AVG) is a set of vehicles used for transporting people or goods, such as a car, truck, or bus, that are located, gathered, or classed together and are characterized by constant change or progress within the traffic system.

The focus is on decentralized autonomous vehicle grouping, which allows the flexibility of single vehicles to be retained while also enabling the use of group coordination to achieve higher throughput in urban networks, as already witnessed in highway vehicular platoons. A known and practiced concept for urban traffic control at traffic signals is to bundle vehicles passively according to green signal phases; the novelty being active coordination of the vehicles in decentralized groups of interests. Likewise, AVGs make coordinated decisions with and without communication depending on the similarities of their vehicle properties and destinations. AVGs coordinate the motion of traffic, making strategic (i.e., group destination) and tactical (i.e., speed and gaps) group decisions in a street network.

To support this more dynamic and decentralized approach, traffic control needs to be able to handle autonomous vehicles. Different modeling approaches and methods are needed which must also be adapted to group individual vehicles. This PhD thesis proposes methods of automation for vehicle group formation and operation, and shows how decentralized systems can be configured to adapt to realistic traffic scenarios.

The AVGF proposal has been validated using an agent-oriented traffic simulation system and has been embodied in an implementation using realistic scenarios, allowing comparison between the research hypotheses to real-world analysis. As the main contribution, the creation of AVGs and their implementation within realistic urban networks is shown. This is evaluated in an agent-based simulation tool, where the economic benefit was quantified at 14% traffic flow improvement at rush-hour when decentralized group coordination was used.

This thesis provides a new traffic concept of AVGF for alleviating congestion in urban traffic, therefore promoting greater traffic flow efficiency.

Zusammenfassung

Diese Dissertation nimmt an, dass autonome Fahrzeuge die Zukunft der Mobilität sind und somit zu mehr Verkehrseffizienz führen. Koordinationsmethoden sind unabkömmlich, um dem Wachstum der Wirtschaft und der Mobilität zu begegnen. Diese Arbeit erforscht die Leistung von autonomer Fahrzeuggruppenbildung (AFGB) basierend auf einem dezentralen Modell und Simulationsauswertung für urbane Umgebungen. Eine autonome Fahrzeuggruppe (AFG) besteht aus einer Zusammenstellung von Fahrzeugen, die Personen oder Güter transportieren. Zum Beispiel ein Auto, Transporter oder Bus werden durch die konstante Veränderung oder Ablauf des Verkehrssystems lokalisiert, zusammengefasst oder klassifiziert.

Die Fahrzeuggruppierung steht im Fokus, um einerseits die Flexibilität der einzelnen Fahrzeuge zu behalten, andererseits auch den höheren Durchsatz in urbanen Straßennetzen durch die Gruppenkoordination zu erreichen - ähnlich der Fahrzeugkolonnen auf Autobahnen. Ein bekanntes und praktiziertes Verkehrsführungskonzept ist Fahrzeuge passiv nach den Ampelphasen für eine grüne Welle zu pulken; die Neuerung ist aktiv die Fahrzeuge in dezentralen Interessengruppen zu koordinieren. Abhängig ihrer Ähnlichkeiten von Fahrzeugeigenschaften und ihren Fahrtzielen tätigen AFG mit und ohne Kommunikation koordinierte Entscheidungen. Mittels dieser strategischen (z.B. Gruppenfahrtzielen) und taktischen (z.B. Geschwindigkeit und Abstände) Gruppenentscheidungen fahren AFG in koordinierter Weise im Stadtverkehr.

Für die Unterstützung des dynamischeren und dezentralisierten Ansatzes ist zuerst eine Verkehrsführung für autonome Fahrzeuge und schließlich verschiedene Modellierungsansätze und -methoden für das individuelle Gruppieren von Fahrzeugen notwendig. Diese Dissertation schlägt Methoden zur automatischen Fahrzeuggruppenbildung und -ausführung vor und zeigt auf wie diese dezentralisierten Systeme konfiguriert werden, um sich realistischen Verkehrsszenarien anzupassen. Der AFGB-Ansatz wird durch eine Implementierung mit realistischen Szenarien umgesetzt, die einen Vergleich der Forschungshypothesen und deren Analyse erlaubt, und mit einem Simulationssystem validiert. Als Hauptbeitrag werden AFG mit einem eigenen agentenbasierten Simulationswerkzeug realisiert und evaluiert. Neben anderen Erkenntnissen zur dezentralen Gruppenkoordination wird der ökonomischen Nutzen als Verbesserung des Verkehrsdurchsatzes um 14% im Berufsverkehr beziffert. Diese Doktorarbeit bietet ein neues zukunftsfähiges Verkehrskonzept durch AFGB.

Acknowledgements

Completing this thesis at Clausthal University of Technology (TUC) in Germany seemed to be a long way. More often, however, it was a very enjoyable experience, especially because it came with the opportunity to meet and work with many great people. Thus, all these people contributed in the one or the other way to the thesis. I take this opportunity to thank everyone who helped me completing this challenge! A few people to which I am especially grateful, I would like to name explicitly in the following.

First and foremost, I am very thankful for the support I got from my supervisor and friend Professor Dr. Jörg P. Müller. Jörg on one hand provided me with lots of knowledge of agent-based systems and on the other hand with the confidence that I am on the right track while at the same time never stopping in thinking ahead and demanding more. His suggestions and skills of describing ideas helped me a lot in achieving the solutions presented in this thesis. Jörg's feedback not only contributed to raising the quality of the publications on which the thesis is based, but it helped me in assessing research quality myself. Thanks, Jörg!

Second, I am deeply grateful to my co-supervisor Professor Dr. Sven Hartmann. Prof. Hartmann is an expert on the field of data modelling and development, algorithms and databases. Thank your for being my co-supervisor Sven!

Furthermore, I am very grateful for being part of the PLANETS team. In particular to the IVS team of Prof. Dr. Bernhard Friedrich for providing their realistic model and data for the case study, as well as to the institutional teams of IKT of Prof. Dr. Markus Fiedler, BIS of Prof. Dr. Dirk Mattfeld and MEC of Prof. Dr. Jörg P. Müller involved in the case study for their input and opinions. This research has been partly supported by the German Research Foundation (DFG) through the Research Training Group SocialCars (GRK 1931).

Thanks to the invitation by Sascha Ossowski, I was able to visit URJC in Móstoles (Madrid) in Spain and the artificial intelligence group. Not only I very much appreciated their feedback I received, but I was introduced to new friends and a scientific community, especially being involved in the European Cooperation in Science and Technology (COST) Action project, with whom I started a very fruitful collaboration. One notable outcome of the cooperation is the book of 'Agreement Technologies' [267] edited by Sascha Ossowski.

Chapter 6 Urban Traffic Groups - A Case Study is the result of a great collaboration with Thomas Hornoff. Therefore, I would like to thank Thomas for implementing and testing the ideas developed in the earlier chapters as the case study's software engineer and thus for giving me extensive feedback on the developed tools and techniques. Also, I appreciated the discussions I had with Jan F. Ehmke, Hugues Tchouankem, Henrik Schumacher, Daniel Schmidt, Maksims Fiosins, Gianina Homoceanu, Michaela Huhn, Viet Hung Chu, Michael Köster, Tristan Behrens, Nils Bulling, Philipp Kraus, Tobias Wessels, Thomas Plathe, Shaghayegh Mehrabieh, Nils Armbrecht, Sophie Dennisen, and Malte Aschermann on various subjects of the thesis a lot.

During the years of my thesis, I enjoyed working in the MEC group of TUC very much. Especially, I would like to thank Christopher Mumme and Sabrina Wittek with whom I shared a very pleasant working atmosphere in our office, characterized by the readiness to help each other and discuss research-related and non-related issues in both a productive as well as in a not so serious manner.

I guess, never I would have managed to cope with all the administrative and personal issues if I would not have had the support of Stefanie Cronjäger, Christine Kammann, Sandra Karpenstein, Andrea Behfeld, Anita Seiz-Uhlig, Thomas Bravin, Peter Platzdasch and Jörg Körner for which I am very thankful. Also, thanks to Patrick Stiefel, Stefan Kehl, René Fritzsche, Sascha Lützel, Jens Drieseberg, David Mainzer, Sven Arnold, and all the other current and former members of the IFI group(s) in Clausthal for the great time during meetings, lunch, chatting, sports etc.

To focus on a single subject over years - as it is needed when writing a thesis - would not have been possible for me if I would not have had friends that made me focusing on other issues in my spare-time. Therefore, I am very glad for the inspiring times I had with Petra (thanks for the editing suggestions!), Simone, Petoula and Nico, Christine and Friedrich (thanks for enhancing my illustrations and discussing the analysis and interpretation of results), Katja and Alex (thanks to revising and focusing on the introduction), Ellen Sophie, Donnée and many, many others in the Harz mountains and elsewhere. Thank you all!

I am very thankful for the support and feedback I received from my family in Germany Görmer's and Masche's, in Australia the Redding's and in the United States the Simcoe's, but especially my sister Katja and my parents. Katja, you are always a good advisor whose feedback I appreciate a lot. Mutti, Vati, I know you are the two who are the proudest of all of this thesis and without your endless support and love I would have never been able to achieve this.

Last but for me most important I appreciate the dedicating support and love from my husband Guy and all the unconditional love, fantasy and distractions of my beautiful daughters Laura and Louisa who gave me the precious time which I needed to finish my thesis project. Thank you so much!

Contents

List of Figures

List of Tables

Abbreviations

2APL	BDI-based agent-oriented programming language, called A Practical Agent Programming Language
2D	two-dimensional geometric model
3APL	Abstract Agent Programming Language or Artificial Autonomous Agents Programming Language, an extension to 2APL
3D	three-dimensional geometric model, an extension to 2D
3T	A leading developer of electronics and embedded systems
A&A	Agents and Artifacts
Aaladin	A meta-model for the analysis and design of organizations in multi-agent systems
AAOL	A meta-model: Autonomous Agents in Organized Localities
AC	Active Component
ACC	Adaptive Cruise Control, this means ”adaptive speed regulation”
ACL	Agent Communication Language
AEDF	Adapted Euclidian distance function for vehicle grouping algorithm
AEIO	Agent, Environment, Interaction, and Organization scheme for this thesis
A-F	range of lot size regarding the LOS
AGENT0	A simple agent language and its interpreter
AHS	Automated Highway Systems designed for driver-less cars on specific rights-of-way
AI	Artificial Intelligence, is intelligence exhibited by machines in computer science
AIMSUN	Simulation package of Transport Simulation Systems integrating three types of transport perspectives
aliCE	Object-oriented 3D programming environment

AMAS	Adaptive Multi-Agent Systems Theory
ANA	Agent Network Architecture
Ant libs	Java library and command-line tool, main known usage of Ant is the build of Java applications
AnyLogic	multi-method simulation modeling tool developed by The AnyLogic Company supporting agent-based, discrete event, and system dynamics simulation methodologies
AOS	Autonomous Decision-Making Software™that underlies autonomy using JACK®, C-BDI and CoJACK
API	application programming interface
AplTk	agent programming toolkit
AplTkEx	agent programming toolkit (AplTk) extended for purpose of creating the simulation environment for this thesis
ARTEMIS	pArticle transport, Recombination, and Trapping in sEMiconductor Imaging Simulations
ASL	standard AgentSpeak language
ATCS	Adaptive Traffic Control Systems
ATSim	Agent-based Traffic Simulation System
ATT	Agents in Traffic and Transport
AUTO21	Canadian research initiative for use of cars and innovation
AVENUE	Advanced & Visual Evaluator for road Networks in Urban arEas, traffic simulation model
AVGF	Autonomous Vehicle Group Formation
BALANCE	traffic-adaptive network control, adaptive signal control method
BCC	Behavior Contol Component
BDI	Beliefs Desires and Intentions (Model)
BPMN	Business Process Model and Notation
C/C++	programming language in structured C or object-oriented C++
C2C	Car-to-Car Communication
C2I	Car-to-Infrastructure Communication
C2X	Car-to-X-change (X stands for any) Communication
CA	Cellular Automata model
CACC	Cooperative Adaptive Cruise Control
CAEPIA	Conference of the Spanish Association for Artificial Intelligence
CAM	Standard Cooperative Awareness Messages
CArtAgO	Common ARTifact infrastructure for AGents Open environments for programming environment artifacts

CCL	research group headed by Prof. Uri Wilensky, Northwestern's Center for Connected Learning and Computer-Based Modeling, created netlogo environment
CCR	cooperative collision resolve method
CCTV	Closed-circuit television
C-D	thick or congested range of lot size regarding the LOS
CNP	Contract Net Protocol
CO_2	chemical formula Carbon dioxide, i.e. exhaust gas from vehicles
COLLAGEN	A collaboration manager for software interface agents, especially for Shared Plans
CommPort	Communication Port
CORBA	Common Object Request Broker Architecture (CORBA) middle-ware for system communication
CPU	Central Processing Unit
CRONOS	real-time urban traffic control system
CS	Color Sort algorithm
CTL	Computation Tree Logic
CTPS	Corporate Telephone Preference Service
DAI	Distributed Artificial Intelligence
DARPA	Defense Advanced Research Projects Agency
DB9	Aston Martin DB9 is a grand tourer largely made of aluminum, top speed of 295 km/h and a 0 to 97 km/h time of 4.1 seconds
DCOP	Distributed constraint optimization problem
DGF	Dynamic Group Formation
dMARS	is an architecture: a specification of the Distributed Multi-Agent Reasoning System
DPS	Distributed Problem Solving
DRACULA	A Dynamic Network Simulation Model Based on Multi-Agent Systems
DRIVE III	Dedicated Road Infrastructure for Vehicle Safety in Europe), the EU's telematics research
DTM	Dynamic Traffic Management
DVG	Dynamic Vehicle Group
Dynameq	multi-scale traffic simulation providing a vehicle-based traffic simulation
dynaMIT	a simulation-based system for traffic prediction

EAII	requirements for Environment, Algorithms, Interaction, Individualization
EASSS	European Agent Systems Summer School
EDF	Euclidian Distance Function
EIS	Environment Interface Standard
EJP	EIS Jason Parser
EL	Execution Layer
ETRA	customized development for mesoscopic simulator
ETSI	European Telecommunications Standards Institute
FCD	Floating Car Data
FCM	Factors-Criteria-Metrics-method
FDP	Forward Dynamic Programming
FIFO	First in, first out
FIPA	Foundation for Intelligent Physical Agents
FPS	Frames per second
FSP	Full Shared Plans
FVDM	Full Velocity Difference Model
Gaia	methodology for agent-oriented analysis and design dealing with both the macro-level (societal) and the micro-level (agent) aspects of systems
GAP	Gap Acceptance Model
GB	Giga Byte to quantify computer memory or storage capacity
GeoMason	Geospatial support for MASON
GPS	Global Positioning System
GGM	Generic group model
GHR	Gazis, Herman and Rothery: safety distances of different vehicles in a platoon
GHz	gigahertz clock frequency representing a cycle of time
GIS	Geographic Information System
GLP	Grouping Literal Parser
GM	Generalized model
GoD	Group-oriented driving
Golog	Logical approach(es) like ConGolog or IndiGolog to agent and robot programming
GPGP	Generalized partial global planning
GPMN	Goal Oriented Process Notation, extends BPMN

GPS	Global Positioning System
GR	Group Member
GTI	Grand Touring Injection
GUI	graphic user interface
HDC	High-Definition Coding,
HDF	Hierarchical Data Format is a set of file formats (HDF4, HDF5) designed to store and organize large amounts of data
HLD	High Level Design
I2V	Infrastructure to vehicle communication, like C2I or V2I
ICDP	Initial-Commitment Decision Problem
ICL	Individual Context Layer
ID	Identifier, a symbol which uniquely identifies an object or record
IDM	Intelligent Driver Model
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IM	Intersection Manager
INDRA	Traffic simulation with customized development for mesoscopic simulator
INI	format is an informal standard for configuration files for some platforms or software
InteRRaP	Integration of Reactive Behavior and Rational Planning, an agent architecture for multi-agent systems
IN-TUC	Integrated Traffic Responsive Urban Traffic Control
IO	input or output
IP	Internet Protocol
IQR	interquartile rate
IRMA	Intelligent Resource-bounded Machine
IT	Information Technology
ITS	Intelligent Transportation Systems
ITSA	Intelligent traffic signaling agents
ITSC	Intelligent Transportation Systems Conference
IVT	Institute for Transport Planning and Systems, TU Berlin
JaCaMo	A combination of Jason, CArtAgO and MOISE
JACK®	collection of autonomous agents that take input from the environment and communicate with other agents

JADE	Java Agent DEvelopment Framework
JADEX	a Belief Desire Intention (BDI) reasoning engine that allows for programming intelligent software agents in XML and Java, extension of JADE
JAL	JACK Agent Language
Jason	interpreter for an extended version of AgentSpeak and for programming autonomous agents
JAR	Java ARchive is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.)
JAVA	general-purpose computer programming language that is concurrent, class-based, object-oriented
JCC	controls the loading of JS and CSS files globally, extension for a plugin
JI	Joint Intentions
JRE	Java Runtime Environment
JRep	combination of simulation platforms Repast and JADE
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
log	log file
LOS	Level of Service for Traffic and Speed
LTL	Linear Temporal Logic
MANET	Mobile ad hoc networks, technology for Vehicular ad hoc networks (VANET)
MAS	Multi-agent Systems
MADeM	Multi-modal Agent Decision Making
MADKIT	Multi-Agent Development Kit
MASON	Multi-Agent Simulator of Neighborhoods/Networks
MATI	Multi-Agent Traffic Interface
MATSim	Multi-Agent Transport Simulation Toolkit
MJP	MATI Jason Parser
ML	Mechatronic Layer
MOISE	organizational model for Multi-Agent Systems based on notions like roles, groups, and missions for programming multi-agent organizations
MovSim	multi model vehicular traffic simulator

NCS	non cooperative situations
NP	Nondeterministic Polynomial Time
OC	Organic Computing
OMNeT++	extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators
PAI	Parallel Artificial Intelligence
P2P	peer-to-peer
PARAMICS	Modular microscopic simulation for real world traffic
PLE	Personal License Edition
PSP	Partial Shared Plans
PRS	procedural reasoning system
Repast	agent-based modeling and simulation platform
RoMADS	Robotic MultiAgent Development System
SARTRE	Platooning Program for Public Motorways
SCL	Social Context Layer
SeSAm	Shell for Simulated Agent Systems
SLX	Simulation Language with Extensibility
SP	Shared Plans
SQuaRE	Systems and software Quality Requirements and Evaluation
STEAM	a Shell for TEAMwork: hybrid general model based on the joint intentions theory and partially also in Shared Plans theory
SUMO	Simulation of Urban MObility
SWOT	Strength Weakness Opportunity Threat Analysis
TraCI	Traffic Control Interface
TraCI4J	A high-level Java library to communicate with SUMO through its TraCI protocol
TC	Traffic Control
TM	Traffic Management
TU	Technical University, i.e. of Clausthal or Braunschweig, Germany
TUC	Traffic-responsive Urban Traffic Control, multi-agent control for linear, dynamic systems
UML	Unified Modeling Language

URL	Uniform Resource Locator
USA	United States of America
UTC	microscopic traffic simulation with adaptive control
UTOPIA	microscopic traffic simulation with adaptive control
V2V	Vehicle-to-Vehicle communication, same as Car-to-Car Communication (C2C)
V2I	Vehicle-to-Infrastructure Communication, same as Car-to-Infrastructure Communication (C2I)
V2X	Vehicle-to-X-change (X stands for any) Communication, same as Car-to-X-change (X stands for any) Communication (C2X)
VACS	Vehicle Automation and Communication
VANET	Vehicular ad hoc networks
VII	Cooperative systems with connected vehicles direct linked to the traffic management
VISSIM	PTV tool for transportation planning, traffic engineering and traffic simulation
VISUM	PTV tool for transportation planning, traffic engineering and traffic simulation like PTV tool for transportation planning, traffic engineering and traffic simulation (VISSIM) for transport master plan - overall eye
VMS	Variable Message Signs
VSPlus	Adaptive control interfaces for microscopic simulator
VW	German car producer Volkswagen
WAG	weak achievement goal
WI	world interface
Wi-Fi	technology for wireless local area networking with devices
XML	Extensible Markup Language
ZGZ	traffic simulator with customized developments for mesoscopic simulations

The beginning is the most important part of the work.
Plato (s. IV aC)

Chapter 1

Introduction

This thesis addresses a new model of autonomous vehicle group formation (AVGF), which will benefit the flow of urban traffic in the future. Increasing population and cross-linked economies with incremental division of work trigger a growth in transportation processes and raises the question of where traffic management needs new constructive concepts. Passenger mobility has had a long tradition in our industrial society. Figure 1.1 illustrates the historical development and new modes of transportation such as rail, automobiles and aviation which have had impact on transportation. Self-driving vehicles are the (future) robots to simplify human lives and the next transformative technology, as once the introduction of the automobile, the traditional car, was.

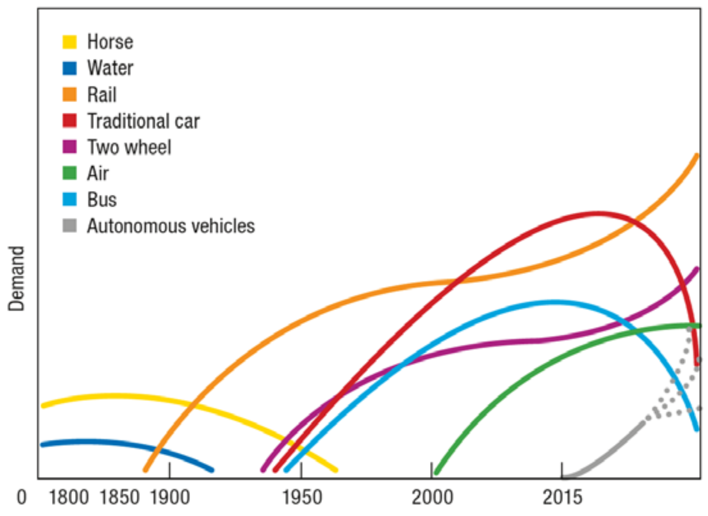


Figure 1.1: Indication of Historic and Future Passenger Mobility Trends (cf. [295]).

The news [275] [54] [311] [330] [58] insinuates [99] that the future of mobility lies in highly or fully automated vehicles. In the upcoming years, highly assisted vehicles (HAV) will be on the market, whereas fully automated cars are under active development and topical in transport research. Nevertheless, the development of self-driving cars that do not require human intervention, also referred to as autonomous vehicles (AVs), is accelerating as described by `Bertoncello and Wee` [41]. They discuss that, in 2015, the development of fully autonomous vehicles starts, in 2020 AVs are market-ready, and in 2030 the consumers start to adopt AVs. Finally in 2050 AVs become the primary means of transport.

The classification of automated driving levels provided by the Society of Automobile Engineers (SAE) [181] proposes six levels, starting from level 0 (purely manual driving) to level 6, where the automated system can perform all driving tasks under all conditions. The United States Department of Transportation applies the SAE levels with National Highway Traffic Safety Administration (NHTSA) [2] policies: On the one hand, it includes guidance for manufacturers, developers and other organizations for designing, developing, testing and deploying safe automated vehicles. On the other hand, it provides policies with a clear distinction between Federal and State responsibilities and existing and new regulative tools required for AVs. European mobility adoptions deal with automated driving [72] including various functions for daily traffic. There is a European study [135] which considers highly automated and connected vehicles and the necessary policies for sustaining research and development and bringing them to market.

Besides the above-mentioned technical and political achievements regarding autonomous vehicles, there are technologies and systems such as Cooperative Adaptive Cruise Control (CACC) and Intelligent Transportation Systems (ITS) in use. ITS aims at improving public transport, logistics and traffic management by utilizing new hardware infrastructures such as sensors and communication networks with modern information technologies. Examples of ITS are presented in different projects like PATH [354] or AUTO21 [238]. One important research area of ITS is the construction of intelligent autonomous vehicles, such as one driving in the competition of DARPA Grand Challenge or the construction of the Google Car 'waymo'[356], which are able to collaborate with each other to reach their destination safely and efficiently. CACC provides vehicles with the ability to cooperate, e.g., in order to avoid collisions. Collaboration in this context is the coordination and cooperation of individual vehicles in order fo them to reach their goals efficiently (further definition of keywords in the following Chapter 2).

1.1 The Argument

Autonomous vehicles (AVs) have recently been receiving much attention due to their promising prospects of social, political and economic benefits. AVs are capable of sensing their environment and navigating without human input. AVs interact with smart infrastructure and communicate with each other. The main features of AVs are the capability of autonomous operation like the Uber autonomous taxis [158], autonomous delivery of goods and persons [55], as well as automatic parking [217]. Here, autonomous indicates that the vehicles act independently of the driver's action and control, whereas automatic is defined as working by itself with little or no direct human control, but still initiated by the human.

AVs will act as individual robots, driving the demanded route using calculated decision making, while travelers will be able to sleep, eat, email, work or even meditate. AVs can be expected to become places of activity rather than just means of transport. AVs have many possible uses in the transportation field [390] [131] [250], including logistics, such as the trial of autonomous pods for home delivery [55] and also in economic use to maximize the safety and efficiency [249].

However, AVs are still heavily under development and manufacturers are more or less successfully striving to establish them on the market [2]. The rising technological standard [244] and increasing amounts of traffic burden the existing traffic management. New urban planning concepts and new technologies such as V2X (vehicle to exchange communication) and autonomous cars are research in progress and only rudimentary available.

Standard traffic situations include numerous heterogeneous participants (here: vehicles). There are many situations where communication between participants is beneficial, if not crucial. An example is a traffic jam, in which no individual vehicle can improve the situation by itself, other than by adjusting their speed to the conditions and waiting until the cause of the traffic jam has dispersed. In congested situations, information such as warnings to slow down and problem solving techniques for resolving the cause of conflict is helpful. Grouping vehicles as investigated in this thesis could be beneficial in order to increase the throughput of a network, but at the same time focus on the individual interests of vehicles to increase the driving comfort. The specifications of AVs are designed by engineers based on the model of human driving and decision making. Thus, AVs can have goals and interests.

Therefore, this work focuses on the investigation of how autonomous vehicles can be grouped and what effects such grouping may bring about in different traffic situations. One way to model AVs is in simulation [281] [215] by incorporating AVs into vehicle groups. Industrial fleets lead the way with the use of grouped AVs in mining, farming and in closed contexts like airports.

Two relevant concepts in this context are convoys [38] and platoons [354], where vehicles are joined for coordinated action. Convoys are a sequence of vehicles driving in the same lane in which every vehicle has a driver (usually they are military or logistical convoys). Vehicle platoons are an automated form of convoys in which a human driver leads a line of closely following vehicles. Each following vehicle autonomously measures the distance, speed and direction and adjusts to the vehicle in front. In more recent works `Saeednia and Menendez` [301], the term 'platoon' is defined by decreased gaps between consecutive trucks and identical speeds.

Automobile platoons are used for automated highway systems (AHS) only. It started with the first automated vehicle, which contained a computer and was built for research by Ohio State University in 1962. For implementation in the PATH project of automated vehicle platoons, the infrastructure needed to change, because it works with magnetized stainless-steel spikes in the highway which provide small amounts of digital data (describing lane changes, recommended speed). Vehicles use this infrastructure-provided information for sensing speed and their location. Thus, vehicles can organize themselves into platoons of eight to twelve cars and decrease the distances between each other to a conventional braking distance. Although the platoons were successful, investment has moved towards autonomous intelligent vehicles rather than building specialized infrastructure.

Nowadays, the AHS[1] platoons are conducted by the SARTRE Project (Safe Road Trains for the Environment) [67], which includes sensory technology in vehicles (mostly trucks) that can read passive road markings, and use radar and inter-car communications to make the vehicles organize themselves without the intervention of drivers. Being in the platoon, drivers can do other things than driving while the platoon proceeds coordinated towards its long distance destination. The vehicles in the platoon are physically detached and can leave the convoy at any time. Such an autonomous cruise control system has been developed by Volvo, Mercedes-Benz, BMW, Volkswagen and Toyota.

However, although the effect of vehicle platoons on highways was demonstrated (benefits of substantially shorter commutes during peak periods, reduced congestion, fewer traffic collisions, and greater fuel economy due to reduced air resistance) [67], in general little attention has been paid to the use of grouping vehicles in urban traffic.

1.2 The Example

One sample traffic scenario models traffic components in a four-way intersection as seen in Figure 1.2. The focus is on the individual vehicles, seen as autonomous agents (explained in Chapter 2.3.1) which coordinate their calculated behavior with other agents. The agent concept includes reactive, proactive and social behavior. Exchanging information pertaining to goals and route plans is essential for decentralized cooperative control in dynamic groups of vehicles.

From the point of view of agent design, this application example is appealing because, on one hand, it is easy to understand, while on the other hand, it offers a variety of interesting problems. The local planning of groups and the necessity of dealing with changes caused by the actions of other agents allow us to study the relationship between individual and collective group vehicles. The agents in urban traffic have to react in real time, they are resource-bound, and have incomplete knowledge about the world.

The main focus of this thesis is the integration of agent coordination and cooperation into urban traffic environments in the form of groups. Use cases of traffic scenarios make it possible to investigate various forms of vehicle grouping strategies using a variety of coordination and cooperation mechanisms. The mo-

[1]There exists a broad line of work on automated highway systems like the concept of managed lanes.

tivations for forming a vehicle group are less gaps, higher speed, less emissions, and higher traffic safety.

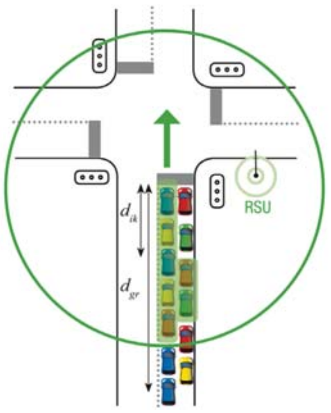


Figure 1.2: Vehicle Grouping (adapted Source: [110] p.12.)

For example, the mini scenario in Figure 1.2 illustrates a typical situation. There are two lanes in each direction with three vehicle options for the next action: turn left, go straight, and turn right, which would cause a resource conflict for lanes; thus, three lanes in each direction and three options to take means that preprocessing of the route plan is necessary. The situation needs vehicle coordination before arriving at the stopping line. Obviously, there are different ways of resolving those situations.

One method of vehicle group formation is illustrated in Figure 1.2. The first vehicle (blue in the illustration) at the stopping line is defined as the group leader. This group leader sends a speed recommendation, the maximum possible group size and information about the destination of his route. The maximum group size d_{gr} (the green area within the green communication circle) is measured as the spacial distance to the group leader and is determined by the predefined desired speed $v_{desired}$ and the rest time value t_{green}: $d_{gr} = v_{desired} * t_{green}$.

Potential group members which are within the communication coverage distance (blue circle) verify with transmitted information whether or not they are joining the group leader. For this purpose, the destinations of the routes need to coincide. In order that all vehicles within a group can pass the stopping line during the next green phase, the distance d_{ik} between the vehicle i and the group leader k should be within the maximum distance d_{gr}. Consequently, the group size is limited by the communication range, the rest time value of the traffic light and the predefined desired speed. Finally, all group members adapt their speed to the recommendation of the group leader (the movement is represented by the green arrow).

Infrastructure elements like traffic lights and message signs can help to control and coordinate vehicle actions or can even be extended by building another lane. But infrastructure changes are not always beneficial or possible. Thus, coordination is the key. For example, the vehicle agents could - each by itself - locally decide for some random action movement. Alternatively, the autonomous vehicles could communicate their goals, and agree on a joint plan to resolve resource conflicts (like one lane for two actions) or for coordination (like sorting depending on the route the vehicle agents have planned). Intuitively, the former option seems to generally be used to deal with the situation, whereas the latter alternative is more reasonable for cooperative traffic control and can bring benefits to the traffic system. Typically, this results in coordination and communication costs which should be minimized by the design of how vehicles are grouped. Vehicle groups should not use extra resources like a commuters' park and ride lot, and communication should be standard and kept to a minimum.

Do simulation tests confirm the intuition that grouping vehicles by route choice brings benefits to urban traffic? How will a traffic system consisting of a number of vehicles behave if the individual agents use group strategies? These are central questions that this thesis investigates.

1.3 The Approach

This thesis tackles the relevant research question of using AVs for a new mobility model by designing and implementing autonomous vehicle group formation (AVGF). In this work, the focus is on AVGF for fully autonomous vehicles (SAE Level 5) exclusively. Since fully autonomous vehicles do not exist, simulations are used to evaluate the vehicle groups. In an urban simulation environment, AVGF joins individual vehicles dynamically into vehicle groups based on their similarities to the surrounding ones. The goal is to prove that AVGF will contribute to a smarter, faster and more efficient mode of transport.

In this thesis, the procedure model of AVGF is addressed in three steps:

1. Functional Design
   In this phase of work, the entities in the traffic domain, i.e., traffic participants, especially vehicles, are defined. Then, the requirements for the simulation of traffic with focus on AVs are analyzed. Simulations are required for modeling AVs in urban traffic, because manufactured autonomous vehicles exist only as prototypes and are still under development; there are legal aspects as well. But simulations have difficulties reproducing interactions, i.e., the communication between drivers and other drivers or pedestrians. Humans often interact with eye contact or gestures and have many psychological influencing factors which are difficult to model. Due to the communicational and cognitive limitations in simulations, the model is simplified to use exclusively autonomous driver-less vehicles. Additionally for SAE level 5, human drivers are the users of the fully automated cars and are no longer active.
   Cooperation (defined in the Background Chapter 2.1.1) is a strategic approach to meeting the challenges of open complex systems. But this will be no universal solution: the choice of strategy depends on the individual situation, although it can be motivated by the infrastructure. AVGF is addressed in this thesis with cooperative coordination methods applied to dynamic environments. Cooperative traffic is considered in this thesis

under the assumption that AVs are cooperative and not competitive in the sense of economics. This means that the vehicles react as rational actors (not altruistically, but tending to be benevolent), considering their own preferences. The global goals are reached through their diametrically opposing group behavior . Malicious behavior or competitive aims are out of scope.

This thesis aims to use the paradigm of intelligent autonomous agents for grouping vehicles. The agents can interact with each other in a virtual simulation environment in an urban traffic context. These dynamic, interactive environments pose interesting challenges for research on specialized capabilities as well as on the integration of these capabilities.

With the paradigm shift [258] from static and centralized traffic systems to dynamic and decentralized traffic systems, modular subsystems can be designed and simulated with the focus on user optimum. The system optimum is not the focus of this research. The user optimum is challenging because, usually, the individual interacts with systems on that level. Individual goals and preferences can be respected by the design of vehicle groups. Then both internal and external validations of group preferences can be discussed. Individual preferences need to be clustered into groups. Then, statements about the system can be formulated, deriving from the decentralized perspective. This enables groups to fulfill and interact with the requirements of the system as a bottom-up approach.

Groups are psychologically inspired. During the development of human society, group formation was a helpful factor. Group formation phenomenons like termites building their society are transferred to technical systems as defined by socionics and bionics in order to investigate whether groups are also useful in the complex traffic domain. Group behavior facilitates information sharing and problem solving. There are two possible reasons why groups are formed. Firstly, joint tasks can be performed in groups, while, alone, they might not be completed. Secondly, each individual can reach the goal, but in a group it can be reached faster, better and/or with less effort. Groups are triggered in certain situations and the members have corresponding defined relations depending on the joint goal and task.

The idea is that, with a designed vehicle group model, the autonomous vehicles will drive cooperatively in groups with less gaps between vehicles and coordinated operational (e.g., speeds), tactical (route) and strategic (destination) goals, depending on their individual and group utilities. The vehicles are detached and can join or leave the group at any time. A group leader depends on the application and mechanism. Groups can have leaders which are dynamically elected; roles and their assignment within a group may change over time. A horizontal group concept is preferred, which needs to deal with group knowledge and the communication between members.

I define cooperative, decentralized traffic management as the processes of monitoring and optimizing network flows, taking the autonomy of networked traffic participants and communication between them (Car2X technologies) into account.

2. Technical Concept
   In this step, I study the applicability of multi-agent models and methods of group formation and coordination with the focus on the requirements and constraints of decentralized traffic management, which means taking individual preferences of traffic participants into account.
   The paradigm of agents, which derives from the field of artificial intelligence, can contribute tothe solution design of autonomous vehicles. According to `Russell and Norvig` ([300] p. 34f.), in the context of Artificial Intelligence (AI), an intelligent agent is

   > an autonomous entity which observes through sensors and acts upon an environment using actuators (i.e. it is an agent) and directs its activity towards achieving goals (i.e. it is ”rational”, as defined in economics). Intelligent agents may also learn or use knowledge to achieve their goals. They may be very simple or very complex. (...) Intelligent agents are often described schematically as an abstract functional system similar to a computer program.

   Most definitions of intelligent agents emphasize their autonomy, whereas `Russell and Norvig` [300] consider goal-directed behavior as the core of intelligence. For my understanding, I combine the autonomy concept with, especially for agent groups, goal-directed behavior. Agents are autonomous computer programs carrying out tasks on behalf of users. They are defined by the three main attributes of being

   - reactive - meaning that they perceive the environmental context in which they operate and take action appropriately depending on it;
   - proactive - meaning that they adapt to change and react to future situations; and
   - social meaning that they are capable of interaction, communication and coordination; they may collaborate on a task.

   Autonomous programs used for operator assistance or data mining are called ’intelligent agents’ as well, but does not match with my understanding described above.
   The agent paradigm extends into economics and cognitive science, as well as interdisciplinary socio-cognitive modeling and social simulations, which makes it a useful model for cooperating autonomous vehicles. Therefore, AVs are conceived to be operated by software agents.
   Multi-agent Systems (MAS) combine AI methods with distributed systems and object-oriented programming. The assumption by `Müller and Pischel` [255] is that, with this MAS paradigm, it is easier to build systems which have the properties of agents.

MAS [268] are a paradigm for constructing complex, software-intensive systems, providing multiple methods for modeling and simulating. The complexity of computer processes in autonomous vehicles requires modeling and simulation methods for design and high level abstractions. Those new developments for software engineering are provided by multi-agent systems [136].

3. Implementation and Evaluation
   The prototypical implementation is carried out in Multi-agent traffic interface, called MATI, an agent-based simulation environment created in this thesis for modeling AVGF, by using scenarios which reflect realistic traffic situations. For instance, the urban network of Southern Hanover was analyzed, modeling morning rush-hour traffic. This combination of simulation and a vehicle group model form the novel aspects of this thesis in which the experiments in different urban traffic scenarios show significant improvements in traffic flows through the use of AVGF.

   After the group operation, the last process verifies the model with the assessment of the simulation data. The effects of the grouping models using selective routing for physically coordinated driving are investigated regarding the question of whether it is better for the vehicle to drive individually or in a convoy of vehicles with or without group formation on the same route in urban traffic. What are the influencing external and internal factors for manipulating the convoy of vehicles and their group coordination?

1.4 The Contribution

The paradigm shift to dynamic traffic control including autonomous vehicles is likely to take place in the near future. One part of the solution to address the challenges is the approach of autonomous vehicle group formation (AVGF): It models autonomous vehicles in the urban traffic environment, uses the multi-agent paradigm to describe capabilities in order to model driver behavior in an abstract yet realistic way, and simulates the vehicle groups in the urban networks for investigations. The focus is on studying local decision-making that takes individual preferences into account, as well as on sharing and distributing information.

This thesis makes two main contributions. First, it describes and specifies agent-based traffic simulators, resulting in a combination of an open-source traffic simulation environment with the MATI framework which applies the agent paradigm to the actors in the simulations. This simulation tool is designed for the general purpose of connecting exchangeable environments with multiple interpreters acting in the same environment. Specifically, MATI is designed for modeling vehicle group formation along with metrics for measuring the efficiency of the resulting grouping algorithm. This AVGF algorithm is implemented and tested in simulation. Second, this research proposes a multi-agent-based architecture and methods for AVGF, and shows that this combination dramatically outperforms an individual driving strategy in simulation.

The desired outcome is to show that street capacity utilization can be improved in rush hours (medium traffic density: Level of Service C or D) with decentralized vehicle groups in order to increase the throughput for more efficient urban traffic. This is done by adjusting the distance and speed within groups. The concept of agents and MAS enables the representation of traffic systems including traffic-independent signal plans for traffic management and Vehicle-to-X-change (X stands for any) Communication, same as C2X (V2X) communication to show the effects of decentralized groups in urban traffic. The hypotheses are stated in the Evaluation Chapter 7.1.

Therefore, vehicles are abstracted as agents that form groups inspired by human group behavior. Thus, the main objective is to study the benefits of AVGF. A realistic example shows the group phases and the effect that coordinated and cooperative driving has, namely, to improve traffic-dense situations in urban traffic. Coordinated operational (e.g., speed), tactical (route choice) as well as strategic (actual position in relation to the destination) criteria for autonomous grouping are the subject of investigation in order to answer the question of if and how vehicle grouping works in simulations.

In summary, the use of AVs in groups are for safety and comfort reasons, but, mainly, to improve the group efficiency for the benefit of individuals and global preferences thanks to the effect of collective knowledge. Autonomous vehicle groups will result in smoother, faster traffic flows, reducing and avoiding congestion. Additionally, vehicle groups promote efficient and smooth acceleration and enable optimal energy use and reduced emissions. The inherent safety of AVs reduces requirements for heavy protective equipment for coping with the collisions resulting from driver faults, thus shedding weight on the car body. Once again, this generates lower emissions both on the road and during manufacturing. Autonomous vehicle group behaviors and the related technological changes improve the quality of life by optimizing driving, which leads to better use of road capacity, improved road design and new methods for traffic management, decluttering of urban spaces, easier parking, investments in public transport, taxis and infrastructure, and better management of urban sprawl. Thus, the new method of autonomous vehicle groups alleviates existing problems. In future the concept will offer flexibility, is easy to use, and is applicable to loose and tight formations of various sizes, and to group formation with and without leaders.

1.5 The Structure

Based on the contribution and the overview illustrated in Figure 1.3, the structure of this thesis is outlined in the following. The Figure 1.3 shows that scientific contribution is based on the traffic system environment. The numbers indicate the contributions. In the theoretical chapter, a group model is presented which is materialized by the all-embracing simulation environment. The simulation environment encompasses computerized strategies for cooperative urban traffic with four elements: the functional design, the technical concept, the implementation and the evaluation of group strategies, which matches the procedure model 1.3, combining the last two points. The strategies in the simulation environment enable experiments and the realization in urban traffic, or now in simulation, but, by 2050, maybe in reality.

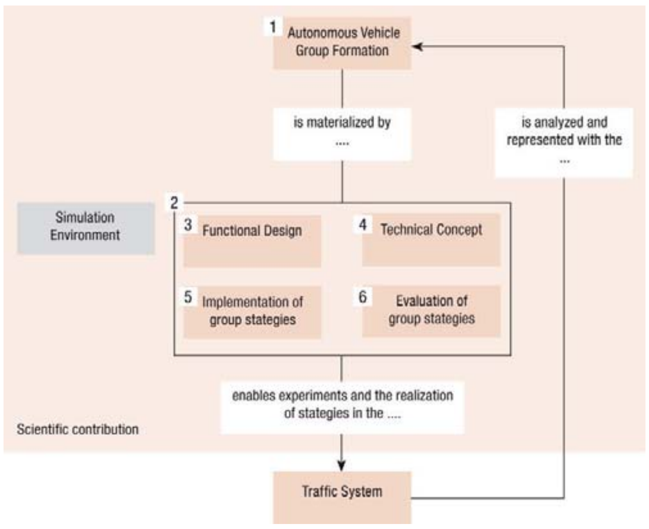

1 Autonomous Vehicle Group Formation
is materialized by ....
is analyzed and represented with the ...
Simulation Environment
2
3 Functional Design
4 Technical Concept
5 Implementation of group stategies
6 Evaluation of group stategies
enables experiments and the realization of stategies in the ....
Scientific contribution
Traffic System

Figure 1.3: Structure of this Thesis.

Chapter 2 (Background) provides the background for this research, starting from the paradigm shift from static and central to dynamic and decentralized systems and the use of a Multi-agent Systems (MAS) approach. The three parent disciplines which influenced this research, Organizational Psychology, Dynamic Traffic Management and Agent Systems, are presented. This chapter provides a brief survey of these disciplines and their relationship in order to place this work in context. Several fundamental modes of cooperative social interaction are described and the rationale behind some basic assumptions of system design is noted. The notion of groups is introduced because of the pivotal role it plays in the definition of the new theory of coordination developed in this research.

Chapter 3 (State of the Art) provides a review of existing multi-agent system traffic applications in order to put this work into perspective. Particular attention is given to the notion of agent models upon which group formation is based. Furthermore, Chapter three reviews existing models of individual agent models and group formation. Prominent models are described and evaluated in order to put the theoretical contribution of this research into perspective. The limitations of using decentralized groups to describe complex systems are discussed. Finally, the failings of existing models of group formation in complex and dynamic environments are enumerated.

Chapter 4 (MATI Simulation Framework) provides an overview of existing agent-based traffic simulation frameworks including own work resulting in the

MATI system, which is described in detail. MATIs novel approach to building multi-agent applications is highlighted, and the shortcomings which led to the group formation work are outlined. The application of MATI to cooperative groups in traffic management is explained in depth.

Chapter 5 (Conceptual Group Formation and Formalization) represents the main theoretical contribution of this research. The model of group formation is defined based on a group reference model. A description of how it circumvents the shortcomings of other models of group formation is given.

Chapter 6 (Practical Implementation of Group Formation) introduces the traffic scenarios in which group formation is tested and how they are implemented. The usefulness of the concepts is illustrated by applying them to the traffic domain.

Chapter 7 (Evaluation) gives an evaluation of the group formation in the MATI framework. Comparisons with other collaborative driving methods are discussed. The algorithms of static color sorting and dynamic similarity are evaluated in different group phases. The different influences of homogeneous and heterogeneous group characteristics are illustrated and compared to the individual default simulation. Two scenarios of homogeneous and heterogeneous vehicle distribution were executed in two realistic locations of Hanover. The grouping algorithm performs in dense traffic. Results show that dynamic autonomous vehicle groups are a future concept for dealing with urban traffic density.

Chapter 8 (Conclusion and Outlook) gives an analysis of key results in whole and future perspective on the work with recommendations for further investigation. The discussion gives an indication of which direction actual research will go in and reflects the scope and limitations.

No thought exists without a sustaining support.
Mel Bochner (1970)

Chapter 2

Background

This chapter provides an overview of other disciplines and their influences on autonomous vehicle group formation (AVGF). Heterogeneity is a basic principle of AVGF in traffic as well as in many other domains and can seen as a problem to be solved or a challenge to optimize different scopes. The three major parent disciplines are introduced in order to place this work into context: organizational psychology, traffic simulation and agent systems. These disciplines offer possibilities to contribute knowledge for enhancing autonomous vehicles, assuming that for each traffic simulation problem a fitting architecture can be found for part of the solution concept. Several fundamental modes of cooperative social interaction are described and the rationale behind some basic assumptions of system design is noted. The notion of groups is introduced because of the pivotal role it plays in the definition of the new theory of coordination developed in this research.

2.1 Organizational Psychology

Group models are well studied in the area of social and organizational psychology and deal with the human-oriented perspective. Since traffic consists of human users and the inspirational aim of this thesis is group work, a background for group-work and cooperation is provided.

The group process is a fundamental human concept which specifies the goals and the structure of the group as well as the progress of the cooperation between its members. The dynamic group formation process is usually described within the context of social disciplines, but is applied in many domains. The question is, how do humans form groups for joint goals and what phases need to be respected until they can perform the actual common task. What are the actions an individual needs to take to be a member of a group? When is it more beneficial to act as an individual or in a group? What are the benefits? Organizational psychology studies why a human performs a certain action and reveals the correlated behavior. Thus, it covers the transition from a goal to an action performed individual layer in a group.

2.1.1 Groups and Cooperation

The concept of groups and cooperation are assumed for this thesis. In the following subsections the psychological descriptions are given.

Groups

`West` [375] describes groups as a fundamental element of our social experience in living and working together. Groups are, in one sense, an organizational response to the increasing complexity of life. Groups potentially provide social support, safety and security, and adjustment to the environment. Within groups, people pursue shared objectives by integrating diverse skills and, ideally, achieve an optimal use of their resources. Concurrently, organizational goals are more likely to be reached.

Groups are formed in the belief that people working interdependently toward shared goals will lead to synergy and therefore the group performance will exceed the sum of the individuals' performance. Also, groups enable organizations to maintain memories in case one member is absent or leaves the group.

According to `West` [375] (p.6) a group can be seen, on one hand, as a small social system imbued with rational, planned task-performance activity to perform a primary task, and on the other hand, as a performing unit to understand how the group activity relates to the affective and social dynamics of the environment. Socio-technical theories offer unique ideas of the autonomy and self-regulation of groups. The control of the group stays within the group in order to coordinate its own task procedures and the transactions with others outside the group. From the socio-technical perspective, autonomy can be enhanced by differentiating inside group tasks and tasks from others within the system.

Though groups can have any amount of numbers, the minimum is generally acknowledged to be three members. The term 'group' generally suggests a number of conditions which must be satisfied. Members of a group have *shared goals* in relation to their interests, values or representations (such as route, desired speed or type of vehicle). Another group characteristic is *social interaction* in order to achieve those shared objectives. A group identity is created through structural elements, where norms and values are accepted. Defining the role of each group member is optional, whereas, in teams, well defined and interdependent roles are necessary.

Teams are one specific kind of group, which have become an important organizational form for working across structures, functions, time, and space. While there are many similarities between groups and teams, there are some important differences in group initiation, bonding, motivation, time span, and resources, which are listed in Table 2.1 combined from various Internet sources.

A team is a specialized group, which shares technical competence and professional knowledge. It allocates resources, maximizing their use, and has no conflict of interest at the time of appointment in fulfilling a goal together. To work in a team is called a teamwork, which is characterized by:

Table 2.1: Groups versus Teams.

Criteria	**Groups**	**Teams**
What is the purpose?	Support and develop the principles, skills and abilities of members in a chosen domain.	Accomplish a project plan that supports the organization's objectives.
Who belongs to it?	Members from one or many organizations, or not affiliated with any organization.	Members of the organization.
What makes members come together?	Self-selection based on expertise or passion	Selected and assigned by management
What is the glue holding it together?	The passion, commitment to, and identification with the chosen cause or knowledge domain.	The organization plan or the project charter.
What is the nature of the activities?	Goals are more self-generated, best if aligned with organization domain.	Tasks should be aligned with organizational interests. Specific goals from the organization, establishing deliverables and deadlines.
How long does it last?	As long as the members have an interest in building the practice and sustaining the community.	Until the project or work is completed.
What are the resources?	Information, knowledge, experience, member commitment and collaboration etc.	person-hours and work resources.

- flexibility in pursuing a goal
- faith in each other
- interaction

In 1981, the renowned psychologist Belbin [33, 34] defined roles in a team which behave, contribute and interrelate in a particular way. Belbin identified that behavior depends on six influencing factors: role learning, external influences, experience, values and motivation, mental abilities, and personality. The accurate delineation of the team roles helps in understanding the dynamics of any management work team.

The implications of teams versus groups is clearly presented in Table 2.1: teams are are made up of members of an organization and accomplish a plan that supports its objectives, whereas groups are made up of either members from one or many organizations, or not affiliated with any organization, and support and develop the principles, skills and abilities of the members in a chosen domain.

By the joint presence of the following attributes based on Hackman [150] , groups are defined as social entities , that is, as a performing unit embedded in a larger social system (i.e., organizations, here, the traffic network). Groups are formed of individuals with interdependencies who perform one or more tasks relevant to their mission. Their task performance has consequences that affect others (inside and outside). Finally, their group membership is identifiable within and outside of the group.

Cooperation

Cooperation is considered by West [376] (p.3) in terms of its outcomes. It occurs in the context of work relationships among people so that their tasks are achieved successfully. Often, research focuses on the interaction of collaborations. But the thought of cooperation is constructive and pro-social in its interactions. West [376] (p.3) defines cooperation in the following quote:

> Cooperation involves helpful, supportive, and integrative actions that in turn help the team [or group] succeed at its task and strengthen interpersonal relationships.

The interdependency of individual and group beliefs is defined as cooperation by the social psychologist Morton Deutsch [84]. This perspective integrates goal interdependence, interactions and the related outcomes. A cooperative situation stimulates an individual to strive with others for a goal object which can be shared equally among all of them. According to Deutsch[84], cooperation is striving for the same shared or complimentary goals, as opposed to competition, which is striving for the same goal, which cannot, however, be shared. Individuals on one social level are prevented by the rules of the situation from achieving the goal jointly when competing and required to do so when cooperating. They perform better in those situations where goals are pursued jointly. They have a relatively large amount of contacts when cooperating and relatively few when in competition. The capability of each individual is restricted by his or her own biological limitations, and the most effective method to overcome these limitations is cooperation. This requires the formation of a group or non-personal purpose.

Cooperation is a social aspect of an overall situation with social factors arising from it. But those factors can be limiting because interaction must first be enabled and the outcomes of interaction can change the interests and actions of those participating in the cooperation. Cooperation depends on effectiveness, which relates to accomplishments of the social purpose, and efficiency, which relates to the satisfaction of individual interests. The minimum requirement for cooperative behavior is neither the physical togetherness nor joint action, but the lowering of egoistic individual demands. Just the diminution of ego-demands may result in free action and an objective situation. In cooperation, the common goal is more important than any personal objective.

The following quotation from `West` [376] (p.3) summarizes the concept of cooperation:

> In cooperation, people believe goals are positively related. They understand their own goal attainment helps others reach their goals; as one succeeds, others succeed. They then share information, exchange resources, and in other ways support each other to act effectively. Mutual expectations of trust and gain through cooperation promote ongoing efforts to support and assist each other (Deutsch, 1962). This promotive interaction results in relationships characterized by positive regard, openness, and productivity.

Transfer to Traffic Simulation

Transferring this to the traffic domain, team members are usually from one propriety company (like BMW), with all their enclosed information structure; whereas this research wants to be open and consider all autonomous vehicles which are able to exchange information and form groups based on their preferences and characteristics.

Actually, groups are already present in nature, for instance in migratory birds flying in a V formation. This is a good strategy for using less energy than when flying alone, and following the direction of the other birds avoids crashes and keep the group together while migrating.

`Gersick` [129] states that traditional group development theory respects environmental influence only within a small range because all group members do the same work in predicted phases where no changes in interaction between the group and its context are included. Applying this to AVGF, this means that the influence of traffic is only present in the initial situation, whereas, during the process of grouping, the vehicles' environmental changes are excluded from direct influence.

`Hackman` [150] specifies that group studies in laboratories have tended to focus on personal and interpersonal variables and either ignore or hold constant contextual variables. This research is based on simulation and the model is also prone to abstractions; it is especially difficult to include communication at its full variety. For this work, communication is limited to the extent necessary for forming vehicle groups and based on standards like Foundation for Intelligent Physical Agents (FIPA) for agents, or Standard Cooperative Awareness Messages (CAM)s for vehicular communication. The variables are weighted in order to adjust their influence on the traffic context.

The work-group effectiveness by `Hackman` [150] (p. 319) puts

> special emphasis to the design of groups as performing units and to their relations with their organizational contexts.

`Hackman` [150] (cf. p. 319) claims that the experimenter in the laboratory has a major contextual influence due to his/her decisions as to where the study will be conducted, how the subjects are chosen and formed into groups, how the group task is selected and assigned to the group, which rewards are chosen and administrated, which group information and resources are provided for their work, and which basic norms of conduct are established for the research setting. This research will make use of the realistic road network of the southern part of Hanover, Germany, with a thick density in rush-hour traffic between 7:30 and 8:30 a.m., and the subjects are autonomous vehicles which are formed into groups. The group task is adjusted tactical behavior such as minimized gaps and adapted speeds, as well as strategic goals such as destination and route choice. The rewards are based on the utility functions (described in Chapter 5.3) of the individuals and groups. Group information is shared in the form of weights and a route choice based on the well-known Dijkstra algorithm is used. Traffic density is fixed for rush-hour traffic in the four scenarios presented in this work. Norms of conduct are derived from the realistic network, as well as feasibility of the simulation tools.

The organization within which a group functions influences group effectiveness (cf. `Unsworth and West` [355]). The following are the specifications for this research:

- Rewards in the group and organization: by utility functions
- Available technical assistance to support the group with its tasks: restricted by the simulation tools
- Supportive organizational climate for both individuals and groups: cooperative traffic is assumed
- Extent of competition: no competition, just self-interested individual participants thinking to maximize their utility.
- Level of environmental uncertainty in relation to the task: environment is generally known to traffic participants by an initial network map, and with their own base of experiences and reasoning.

2.1.2 Group Models

In the following subsections, two models for groups and teams are explained. The Tuckman and Jensen Model inspired the group phases and the Input-

Process-Output Model provided the foundation for the simulation of the group processes.

Tuckman and Jensen Model

The famous phases of the *group life cycle* were first described by psychologist `B.W. Tuckman` [349] in 1965, later, in 1977, `Tuckman and Jensen` [350] updated those phases. The group development has two areas in focus: interpersonal relationships and task activity. `Tuckman's` hypothesis was that, in order to reach effective group functioning, his four-stage model of forming, storming, norming, and performing needs to be successfully navigated.

The stages of the revised model including the closure of a group are illustrated in Figure 2.1 and are described in more detail by `Bonebright` [47].

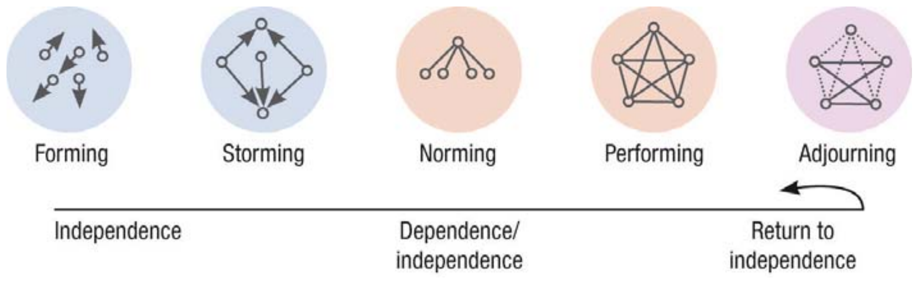


Figure 2.1: Adapted Group Cycle from `Tuckman and Jensen` [350].

Forming: In the first stage, the group members establish relationships with leaders, organizational standards, and each other. The group becomes its task orientation, creates basic rules, and the boundaries for interpersonal and task behaviors are tested. This is summarized as 'testing' and 'dependence'.

Storming: This phase is characterized by missing consensus and emphasis on interpersonal issues. Group members strive for security and therefore resist moving into unknown areas of interpersonal relations. This is due to the desire to express their individuality and resist the formation of a group structure. Groups working toward impersonal and intellectual tasks may have less visible emotional responses, but resistance may still be present. The second stage represents a time of conflict within the group.

Norming: In the third phase, group members accept each other and express personal opinions. Roles and norms are established for developing shared mental models and discovering the most effective ways to work with each other. The group becomes an entity as members develop a group feeling and seek to maintain and perpetuate the group. In an effort to ensure harmony, task conflicts are avoided. During the third phase, the group develops unity.

Performing: In the fourth phase, the group functions as a problem-solving unit. Members adapt and play roles that will enhance the task activities, whereas the structure is supportive of task performance. Roles become flexible and functional, and group energy is focused on the task. In the final stage of the original model, the group relates to their functional roles ([349] p. 387).

Adjourning: The fifth stage, adjourning, reflects separation as an important issue throughout the life of the group. From the return to independence a new group with a purpose and commitment can be established and the group life cycle can restart.

This model was based on a literature review and the observation of a limited number of small group settings. Thus, no attempt was made to establish controls, due to the nature of therapy groups. `Tuckman` pointed out that conclusions about the specific effects of independent variables on group development were missing and encouraged further research along those lines. Limitations of the `Tuckman` model are listed and summarized from `Bonebright` [47]:

- The model is generalized well beyond its original framework.
- Models of groups are limited. Groups were seen as closed systems and not including outside influences on group development.
- The model does not explain how groups change over time. The model should adequately address mechanisms when and how a group moves from one phase to the next and the changes in the group life cycle.
- The model provides a high degree of consistency and similarity for describing the phases as a hierarchical models like Tuckman's. Generally the group development process is more complex which is hard to reflect in linear models.
- There are two significant concerns relating to task performance. The first is that the model fails to address the effects of group development on creativity in problem solving. The second concern is that the model does not guide what tasks are needed to achieve success or what parameters lead to an outstanding performance. This raises two significant questions by `Bonebright` [47] p. 115 :

  > first, what if the storm stage never ends, and second, what is needed in order to exceed performance norms?

- Tuckman's model was studied on therapeutic groups which may cause that especially the storming stage may not be clearly defined. Thus, there might occur limits on the applicability of Tuckman's model outside of therapeutic groups.
- The sample of group development models all fit into a five-stage framework. However, there is variation in the location and definition of the conflict stage.
- Shift to exploring the concerns that drive the conflict, instead looking at conflict as a stage, would cope better with a variety of models found in practitioner literature.
- There is a lack of quantitative research strictly into observations concerning the description and control of independent variables.
- The literature review fails to form(?) a representative sample for small group development processes.

- The settings were significantly overrepresented, particularly the therapy group setting. This limitation has been addressed by further research.

Input-Process-Output Model

Most research and theory in the descriptive tradition shares input-process-output framework by `McGrath` [247].

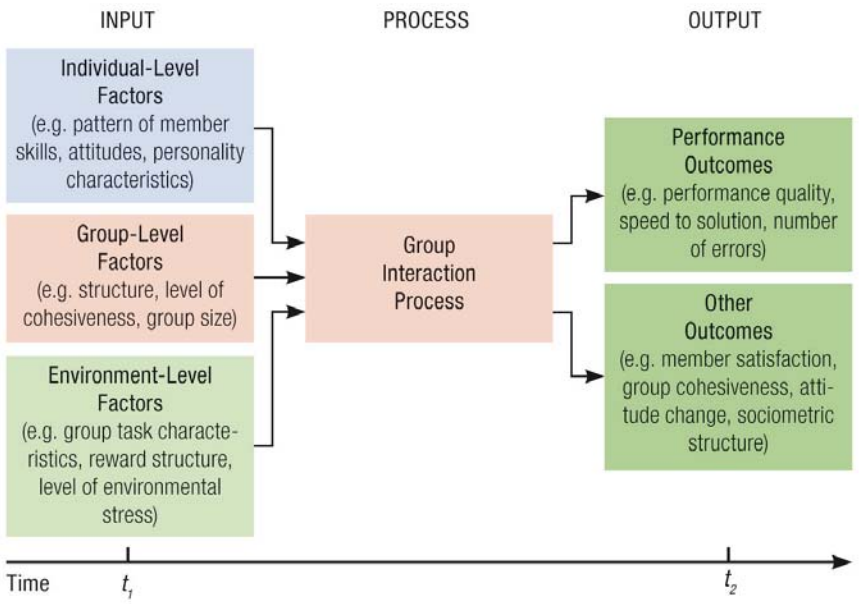

INPUT
PROCESS
OUTPUT
Individual-Level Factors (e.g. pattern of member skills, attitudes, personality characteristics)
Group-Level Factors (e.g. structure, level of cohesiveness, group size)
Environment-Level Factors (e.g. group task characteristics, reward structure, level of environmental stress)
Group Interaction Process
Performance Outcomes (e.g. performance quality, speed to solution, number of errors)
Other Outcomes (e.g. member satisfaction, group cohesiveness, attitude change, sociometric structure)
Time
t_1
t_2

Figure 2.2: Input-Process-Output Framework (cf. originally by `McGrath` [247] and changed by `Hackman` [150]).

For the purpose of this thesis, the input-process-output team model was adopted for groups as illustrated in Figure 2.2.

2.2 Microscopic Traffic Simulation

This section is arranged into three parts: *what* is modeled, *how* is it modeled and *with what* is it simulated.

The ’what’ is three-fold: traffic control in its specific form of dynamic traffic management is provided in Section 2.2.1, traffic flow is the desired improved output with autonomous vehicle group formation and is referred to in Section 2.2.2, and the traffic demand is described through the scenarios explained in Section 2.2.3.

The ’how’ can be approached with macroscopic [216] [154], mesoscopic [216] [200] [312] or microscopic models. Since the focus of this investigation are individual and group vehicle behavior and characteristics, microscopic models were chosen and are presented in Section 2.2.4. There are models that describe the longitudinal behavior, that is the the car following models described in

Section 2.2.4, and there are models which represent the diagonal behavior, that is, the lane changing models identified in Section 2.2.4.

The 'with what' is done in simulation, because autonomous vehicles are not readily in normal use, but under ongoing research. Different traffic simulation software is listed in Section 2.2.5.

2.2.1 Traffic Control

The background knowledge on traffic control is provided specifically for its specialized form of Dynamic Traffic Management (DTM). The input values for microscopic traffic simulation derive from DTM, which reflects the perspective of infrastructure combined with traffic control and traffic information. The architecture of DTM is centralized in a traffic management center, but constant changes need to be made due to the nature of traffic complexity and changing states. Mainly the traffic lights change dynamically depending on the urban traffic. Information is provided for all traffic participants. Improvements of traffic modeling techniques and of computer power promote the further development of sophisticated and complex systems. The general knowledge about this domain resulted from a visit to the Traffic Control Center of Berlin on June 12, 2014, pictured in Figure 2.3, and a presentation by Dr. Kohlen who works there and did the tour. DTM incorporates five main tasks:

1. **dynamic traffic control** of traffic signs and signals, highways, arterials and tunnels
2. **warning system** consisting of an accident committee for traffic security which can include the police
3. **event and incident management** for instance of construction, failures, demonstrations, shootings and state visits
4. **coordination** of public traffic authorities of city districts
5. **traffic information** about all of the above

In the following, the five points listed above are explained in greater detail. 1. Technical systems to support traffic control are mainly the light signal systems commonly known as traffic lights or traffic signals, active traffic management on managed smart motorways, tunnel control, change in direction streets and variable message signs as well as boards. In order to choose the right control strategy,detection can be done with control cross-section and cameras. Fiber optic loops connect the traffic computer with the physical traffic management center of a city. On the mesoscopic level , the traffic computer of a light signal system is responsible for the choosing a signal program and, on the operative level,the light signal system is equipped with a control unit to adapt to signal programs. For instance, in Berlin the traffic management center is connected to nine traffic computers for controlling 2,050 light signal systems.

2. The traffic warning service is connected to the reporting office within the TMC via the traffic message channel and receives current information about

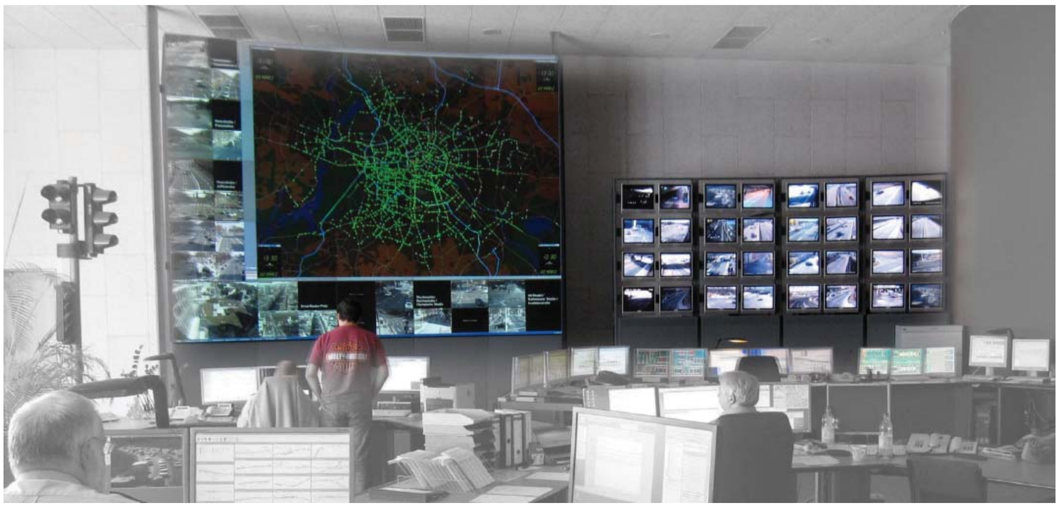


Figure 2.3: Traffic Management Center Headquarters Berlin - provided by [208]

accidents, traffic jams, construction zones or demonstrations. In Berlin there are about 120,000 reports per day (60,000 events and their annulment).

3. The editorial department extracts the information pertaining to city traffic, events and public transport edits and prepares the data for traffic states, information boards and traffic reports. The incoming information is measured in more detail with Traffic Eye Universal for traffic flow, speed and vehicle type, which measures a section of city roads, uses measuring points on highways and measured data from modern light signal systems, as well as Floating Car Data (FCD) provided by TomTom for the network of 1,600 km of roads in Berlin. Current traffic news, construction zones and events, data from public transport, environment and weather data and airport information are also collected.

4. Outgoing information is sent every 5 minutes to support traffic control, create the actual traffic state and traffic news, daily traffic forecasts for media, information boards and statistics of traffic and environment data. A new feature of the traffic information service is the multi-modal mobility monitor with local information about transportation depending on the current position. It provides the actual traffic state in real-time, traffic news, weather, the stops for public transportation with the departure times in a ticker, the position of bike hiring, car sharing and taxis.

5. The information comes from the technical system, the police or the public. All information is received by the reporting office. This traffic information service is responsible, among other things, for giving improved traffic state information to the TMC, for giving traffic and environment data to the administration authorities, and providing information on boards and online for the public and for radio and print media.

Traffic control in the urban environment is necessary due to multiple users which require organization for safety. Current research on co-operative traffic management systems interconnects centralized (system-optimal) control and decentralized (user-optimal) decisions. This requires new dynamic models that take both views into account, and explain as well as predict interdependencies between top-down centralized control and bottom-up local system behavior.

Traffic control models try to influence behavior in terms of changes in the variables, for example the times and costs of travel. Models have evolved considerably to embrace the following planning contexts expressed at various levels of spatial detail (cf. [51] p. 4):

- alternative arrangements of activity locations or land use
- investment in new roads and capacity expansion of existing roads
- investment in public transport systems, and changes to levels-of-service and fares
- traffic management like one-way streets and allocation of lanes to particular users
- demand management like parking restrictions and road user charges
- application of computer and information systems to operate and control transportation systems more intelligently.

This thesis is largely concerned with using existing infrastructure as-is, and focuses on the last point, whereas operational issues such as setting or adaptive control of signals is outside the scope.

2.2.2 Traffic Flow

Traffic behaves in a complex and nonlinear way. Traffic flow [348] is described as the movement and the depending interactions of individual drivers in their vehicles between two points. The unpredictability of driver behavior makes the prediction of traffic flow difficult and without guarantee. Therefore, reasonable consistent ranges of driving tendencies are assumed. Based on assumptions, macroscopic mathematical models are developed. The trends include the relations of speed, flow, and density. These relationships help in planning, design, and operation of traffic networks.

Measurements of traffic density come from infrastructure detectors (buried induction loops, infra-red sensors or cameras installed on traffic lights) or floating car data (FCD). FCD determines the traffic density on the network and collects with help from mobile phones individual data of localization, speed, direction of travel and time information in vehicles that are being driven. Additionally, extra information such as distance to the preceding and following vehicles, fuel consumption and emissions, as well as navigation data such as travel and stop times, sensor data from the environment like temperature, weather and ground, and data from other support systems like ABS or ESP can be used, depending on availability. Aggregation in representation and transmission are known to be existing problems for FCD data. These individual kinds of data are the essential sources for traffic information and for most intelligent transportation systems (ITS). Traffic flow of individual vehicles can be calculated by their travel and stop times.

2.2.3 Traffic Demand

The traffic demand is important to understand. It depends on the time of day and constraints of the environment. In this thesis the traffic demand is set to the yellow C-D range. The technical term for traffic demand is the Level of Service (LOS) and is defined by `Rodrigue` [186] as:

1. A set of characteristics indicating the quality and quantity of the provided traffic service with quantifiable characteristics as well as the ones that are difficult to quantify.
2. A qualitative rating of traffic flow in the range from A (excellent) to F (heavily congested). The actual or prognosticated traffic amount is compared to the maximum capacity of the intersection or street.
3. Public transit agencies in the USA have a variety of measures for the quality of service: generally total travel time or a specific component of total travel time.
4. Furthermore, pedestrian facilities set an area occupancy classifications with levels of service i.e. public toilets.

There are six levels of service as clarified in Figure 2.4:

$$S = sf/(1+a(v/c)^b)$$

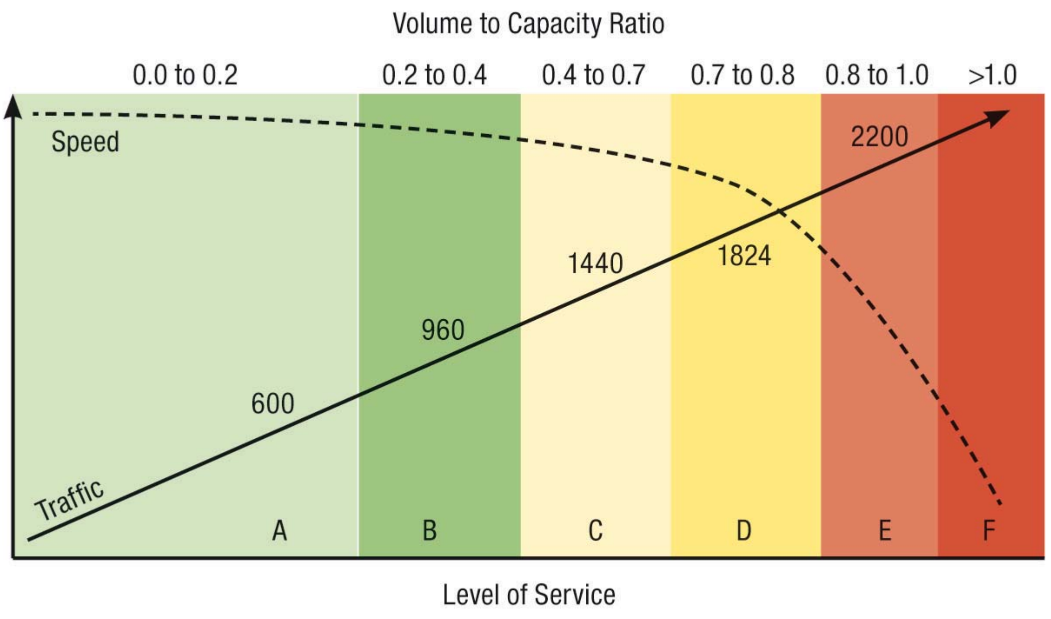


Figure 2.4: Level of Service with linear traffic density. (cf. Transportation Research Board (1994) [43] p. 3-9.)

(A) **Free Flow Traffic**. The presence of vehicles on a road section does not affect other individual users. Speed and maneuverability can be chosen freely. The level of comfort is excellent, as the driver needs to pay minimal attention.

(B) **Steady Traffic**. The presence of vehicles on the section starts to affect the behavior of other individual drivers. Speed can be chosen freely, but the maneuverability has somewhat decreased. The comfort is excellent, as the driver only needs to keep an eye on nearby vehicles.

(C) **Limited Steady Traffic**. The presence of other vehicles affects the behavior of individual drivers and coordination is required. The choice of speed is affected and maneuvering requires vigilance. At this level, the comfort decreases quickly due to the growing impression of the driver being caught between other vehicles.

(D) **High Density Traffic**. Drivers are heavily effected by the presence of other vehicles and there is a need for good coordination. The severe reduction of speed and maneuverability causes a low level of comfort for the driver trying to avoid collisions with other vehicles. Traffic risks increase, causing some operational problems and saturating the network.

(E) **Saturated Traffic**. Drivers are lined up and only a slow but uniform speed is possible. Thus, only under constraint for another vehicle, maneuverability is possible. The driver is frustrated.

(F) **Congestion**. At several points, traffic allows only unstable speed including the formation of waiting lines. Cycles of stop and go with no apparent logic because it is created by the behavior of the drivers. A high level of vigilance is required for the driver with practically no comfort.

Depending on road, traffic and control conditions, the traffic service rate encompasses the maximum vehicles which can cross a point or a road section in an hour. Hence, five traffic rates of service (the F level is not used due to instability) are assigned to each road infrastructure. Traffic reports and digital route information also use color codes to illustrate traffic conditions as follows: green (levels A and B), yellow (levels C and D) and red (levels E and F).

2.2.4 Microscopic Models

Microscopic modeling [216] describes the dynamics of each individual vehicle as a function of velocities and positions of neighboring vehicles. In reality, the driver estimates the distance to the preceding vehicle and his speed in order to avoid collisions and maintain a security distance.

Two important aspects for modeling vehicle entities are the longitudinal and diagonal behavior. Longitudinal actions are deceleration and acceleration and therefore also include following the preceding vehicle. Diagonal action is the lane choice, especially overtaking. In general these two aspects are treated separately.

Car following models are also known as longitudinal models. In the following, different models are presented and compared. Here a preliminary selection takes place as to which models apply to vehicle groups in urban traffic due to their specific characteristics. The implementation of these models is described in Chapter 6.

Car Following Models

Nowadays there are many sophisticated car following models (more are presented in the Appendix) with different approaches and basic ideas which simulate the realistic behavior described. In the beginning, simple models like the kinematic distance model and the model of `Pipes` managed without model parameters like desired and maximum speed, whereas later the models became more complex and developed many different parameters. At this point some conventions, abstract concepts and abbreviations are introduced which all the models presented have in common.

The state of a vehicle in a model is given by its position, its velocity vector and its acceleration vector and is influenced by the driver. In contrast to reality, where the driver of a vehicle influences its state, in simulation different models are used to influence the state of a vehicle. Like in reality, the model needs to react to the behavior of other vehicles. The main focus is on the vehicle in front, the predecessor. Depending on the predecessor's state, that is, his position and speed, the vehicle's own acceleration is adapted.

These descriptions assume a road with only one lane. Velocities are denoted as v_i and positions as x_i (where the index i rises in downstream direction) of all vehicles.

In the following the index n denotes the vehicle ahead and $n+1$ the subsequent following vehicle, as illustrated in Figure 2.5, considering a sequence of vehicles.

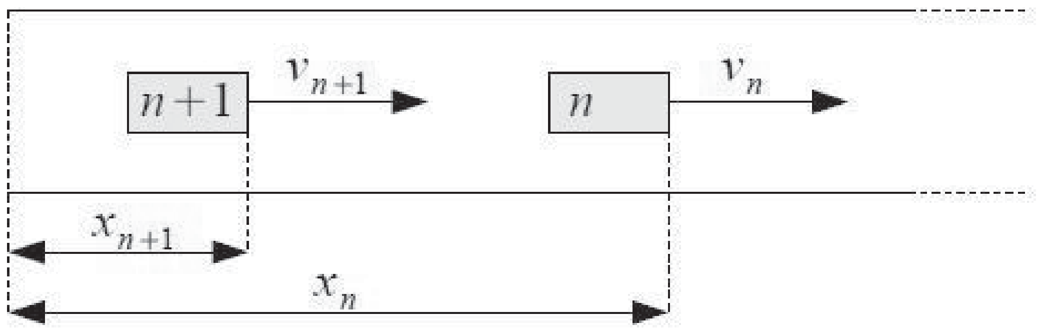


Figure 2.5: Notation of a Sequence of Vehicles.

The velocity or speed of a vehicle n is characterized by v_n. Its position along the driven distance is described by x_n.

The acceleration (or negative acceleration: deceleration) of a vehicle is labeled as a_n. In some models the formula symbol of acceleration is described as the temporal derivative of the speed of a vehicle and therefore given as $\dot{v}_n(t)$.

A change of velocity is performed depending on the desired velocity V_{des}, which is constrained by safety considerations, legal restrictions and other things.

All presented models are continuous in space and time. This is mandatory for the targeted degree of reality. How to profit from the benefits of discrete models is described in 6.

All car following models can be reduced to the simple idea of a relaxation process on some time scale that describes how a driver tries to approach the desired velocity:

$$\frac{\delta v_i(t)}{\delta t} = \frac{V_{des} - v_i}{\tau} \tag{2.1}$$

This dynamic relation is more known for the stimulusresponse approach by `Pipes` [273], where the reciprocal value of τ is referred to as sensitivity.

Different groups of car following models can be identified [284] based on the concept behind the model.

This research considers the safe distance models by `Gipps` [132]) `and Krauß` [216].

Model by Gipps (1981) Traffic flow simulations provide a number of microscopic techniques. Vehicle dynamics and driver decision making is modeled in highly elaborated car-following models. `Gipps` [132] provides the foundation of many traffic simulation tools such as in `Barceló` [21] for a collection of driving strategy models. Those models have automatic cruise control algorithms which resemble vehicle behavior for simulation with their desired speed, acceleration and gaps for lane changes or distances for the preceding car. Agents can incorporate those models. Furthermore, they integrate a high level of decision making and complex information processing is possible.

The model of `Gipps` [132] belongs to the class of models in which the distance between two vehicles is calculated in order to avoid collisions, hence a minimum safety distance needs to be maintained.

In contrast to the `Kinematic` model and models of `Pipes and GHR` (described in the Appendix), where the acceleration is measured, in the model of `Gipps` the speed of a vehicle is calculated directly. The model of `Gipps` consists of two component parts. On the one hand, the speed for free driving is calculated whereby the speed increases in every time step, taking the maximal acceleration into account. On the other hand, the safety distance to the vehicle ahead is calculated including the speeds of other traffic vehicles. This strategy regulates the deceleration or braking that guarantees no collisions. The minimum of both calculated speeds will then be assigned to the vehicle.

The first component of the acceleration results from empirical research according to the equation:

$$v_n(t+T) \leq v_n(t) + 2,5 \cdot a_n T \cdot (1 - \frac{v_n(t)}{V_n}) \cdot \sqrt{0,025 + \frac{v_n(t)}{V_n}} \tag{2.2}$$

with	$v_n(t)$	speed of vehicle n at time t	$[m/s]$
	T	response time of driver	$[s]$
	a_n	maximum acceleration of vehicle n	$[m/s^2]$
	V_n	desired speed of vehicle n	$[m/s]$

The second component of the model regulates suitable braking behavior such that a safety distance is maintained to the vehicle ahead. This includes the reaction time of the following car in the situation so that if the predecessor brakes abruptly, the following car can react.

First, the position where the predecessor $n-1$ would come to a stop if instantly braking at time t is calculated:

$$x^*_{n-1} = x_{n-1}(t) - \frac{v_{n-1}(t)^2}{2 \cdot b_{n-1}} \tag{2.3}$$

with b_{n-1} maximum deceleration of the vehicle $n-1(<0)$ $[m/s]$

The following vehicle n starts to brake after the reaction time T. Thus, the position where the following vehicle stops can be calculated:

$$x^*_n = x_n(t) + [v_n(t) + v_n(t+T)] \cdot \frac{T}{2} - \frac{v_n(t+T)^2}{2 \cdot b_n} \tag{2.4}$$

In addition to the reaction time T in the model by `Gipps` there is a safety reaction time θ. Taking into account the length s of a vehicle, the following must be true $x^*_{n-1} - s_{n-1} \leq x^*_n$ so that collisions are avoided. Inserting the equations 2.3 and 2.4 gives:

$$x_{n-1}(t) - \frac{v_{n-1}(t)^2}{2 \cdot b_{n-1}} - s_{n-1} \leq x_n(t) + [v_n(t) + v_n(t+T)] \cdot (\frac{T}{2} + \theta) - \frac{v_n(t+T)^2}{2 \cdot b_n} \tag{2.5}$$

with	s_{n-1}	length of vehicle $n-1$ plus minimum distance	$[m]$
	θ	safety reaction time	$[s]$

Now the maximum speed at which the mandatory safety distance is maintained can be calculated:

$$v_n(t+T) Re b_n \cdot (\frac{T}{2} + \theta) + \sqrt{b_n^2 \cdot [2 \cdot (x_{n-1}(t) - s_{n-1} - x_n(t)) - v_n(t) \cdot T - \frac{v_{n-1}(t)^2}{b^m_{n-1} ax}]} \tag{2.6}$$

with $b^m_{n-1}ax$ maximum estimation of the maximum deceleration of the predecessor by the driver of the following vehicle $[m/s^2]$

Finally, both components of the model can be combined in such a way that the speed of vehicle n at the time t+T is set as the minimum of both single components. For simplification a lot of applications of the model use $\theta = \frac{T}{2}$. This results in the equation:

$$v_n(t+T) = min\Big\{(v_n(t) + 2,5 \cdot a_n T \cdot (1 - \frac{v_n(t)}{V_n}) \cdot \sqrt{0,025 + \frac{v_n(t)}{V_n}}); \; b_n \cdot T + \sqrt{b_n^2 \cdot T^2 - b_n \cdot [2 \cdot (x_{n-1}(t) - s_{n-1} - x_n(t)) - v_n(t) \cdot T - \frac{v_{n-1}(t)^2}{b_{n-1}^m ax}]}\Big\} \tag{2.7}$$

Krauss Model (1998) In 1998 the pure stimulus-response approach was developed by Krauß [216]. This is a microscopic, space-continuous, car-following model [214] based on the safe speedparadigm: in order to adapt to the leader's deceleration, the follower tries to stay away from the predecessor such that a safe distance and speed can be maintained. As opposed to the IDM A.7, the Krauß model is discrete in time because it does not compute the instantaneous acceleration but the future speed at time step $t + \Delta t$ to be reached by a vehicle i. The model tries to account for human sporadic and irrational reactions with the help of a stochastic parameter τ. The following parameters are used in the model:

	a	maximum acceleration of the vehicle	$[m/s^2]$
	b	maximum deceleration of the vehicle	$[m/s^2]$
with	v_{max}	maximum velocity of the vehicle	$[m/s]$
	l	length of the vehicle	$[m]$
	e	driver's imperfection in holding the desired speed	between $[0,1]$

The model assumes that the driver has a reaction time τ of about one second. The safe velocity is calculated using the following equation:

$$v_{safe}(t) = v_l(t) + \frac{g(t) - v_l(t)\tau}{\frac{\bar{v}}{b(\bar{v})} + \tau} \tag{2.8}$$

	$v_l(t)$	speed of the front vehicle	in time $[t]$
where	$g(t)$	distance to the front vehicle	in time $[t]$
	τ	driver's reaction time	usually $[1s]$

The equation 2.8 computes the speed of a vehicle which is required to maintain a safe distance and avoid creating an accident with the front vehicle. The minimum of these values is computed next, because v_{safe} can be larger than the maximum speed permitted on the street or larger than what the vehicle is capable of reaching until the next step due to its acceleration capabilities:

$$v_{des}(t) = min\{v_{safe}(t), v(t-1) + a, v_{max}\} \tag{2.9}$$

The resulting speed is called the desired speed which is the target speed in 2.9 reached by the vehicle. It is a simple increment from the previous speed limited by $v_{safe}(t)$ and v_{max}.

Assuming the driver is not able to perfectly adopt the desired velocity, the 'driver's imperfection' value multiplied with the car's acceleration ability and a random number is subtracted from the desired velocity. Finally, one must assure that the vehicle is not driving backwards. For this, the last equation of the model is:

$$v(t) = max\{0, rand[v_{des}(t) - \epsilon a, v_{des}(t)]\} \tag{2.10}$$

The velocity, multiplied with the simulation step duration, which is constant at one second, here, is added to the vehicle's current position to reach the position for the next time step. This model is collision-free and very fast due to the small number of equations. For example, on a 1GHz computer the performance of the simulation step for about 1.000.000 vehicles per second can be executed, describing realistic vehicle movements in one second.

The traffic flow is a function of the traffic density acting as the fundamental diagram. The model by Krauß is capable of replicating the flow function well. The Krauß model is used in the traffic simulator SUMO [21].

Lane Changing Models

In microsimulation, the diagonal behavior of individual vehicles is represented by lane changing models. Compared to the car following models 2.2.4 the parameters used are less manageable and depend strongly on the situation. In the literature, lane changing models are addressed much less than the car following models described in Subsection 2.2.4.

Partly empirical and theoretical analysis of lane changing were executed on twolanefreeways and threelane freeways [216]. A model for the structure of lanechanging decisions in urban traffic situations, where traffic signals, obstructions and heavy vehicles all exert an influence, has been developed by Gipps [133, 134].

The lane change models are typically considered as a multi-step process and on a strategic level. The drivers know their route in the network, which influences the lane choice. In the tactical stage, the driver accelerates or decelerates before the intended lane change. In addition drivers possibly cooperate in the target lane [202]. The general lane change begins with the vehicle executing the lane change in the start lane and aiming for the destination lane. The decision to execute a lane change depends on the actual traffic situation, usually on whether it is safe or desirable to change lanes. Additionally, there might be the necessity of performing a lane change. For example, it is desirable to change lanes if the actual speed is recognizably below the desired speed due to a high volume of traffic in the current lane. However, the lane change is mandatory if the vehicle wants to turn soon but is in the wrong lane at present. If the desire or the necessity for a lane change is determined, consequently there is the stimulus for the lane change [133]. The verification of whether this is possible and safe is common to models. The basics for the following two approaches is the *gap acceptance model*, which validates if the gap between the front and rear vehicle in the target lane is sufficient. It must be ensured that the vehicle behind the executing vehicle is unhindered while the lane change is performed. Therefore, the distance between the vehicles should be big enough to include the safety distance and also to avoid making the rear vehicle brake strongly. The same is true for the front vehicle: the distance should be big enough including the safety distance in the destination lane. Depending on the model, those parameters can be different for each driver and traffic situation. The gap acceptance model is a regression analysis [157] where the vehicle decides if the gap is sufficient or not and then accepts or denies it.

In the literature most lane-changing models classify lane changes as either mandatory or discretionary [133] [387] [4] [345].

The well-established typical model by `Gipps` [133, 134] is presented, which is still in use and has been extended by other models. The model `MOBIL`, which seems very interesting for group behavior, is presented thereafter. .

Gipps Model `Gipps` [133, 134] is a standard and presents the Gap Acceptance Model (GAM) with an algorithm. The basic idea is that the driver n changes to lane i, if the following conditions are true:

- The lane i has a gap for a lane change.
- On the lane i the driver n needs to ensure that his follower s can follow him.
- On the lane i the driver n needs to ensure that his future predecessor p can follow him.

The car following models were described before.

The algorithm 1 describes the lane change of a driver n:

Algorithm 1 GAM of Gipps [133, 134]

```
The driver $n$ wants to change to lane $i$
$s, p$ are the following car and the predecessor respectively on lane $i$
$TargetGap$: the distance between $s$ and $p$
if $TargetGap >$ length of $n$ then
    calculate speed $G_m$ of $n$ with a car following model with $p$ as a forerunner
    if $n$ can reduce its speed $G_m$ then
        calculate speed $G_s$ of $s$ with a car following model with $n$ as a forerunner
        if $s$ can reduce its speed $G_s$ then
            $n$ can change to lane $i$
        end if
    end if
end if
```

MOBIL Model The acronym MOBIL stands for the strategy of 'Minimizing Overall Braking Induced by Lane Changes'. It uses accelerations as utility functions and takes the (dis-) advantage of the followers into account via a politeness parameter. Motivations can be diverse from purely egoistic to more altruistic behavior. For example, the value for lane changes is taken into account, only if lane-changing increases the combined accelerations of the lane-changing driver and all affected neighbors [202]. The model uses the Intelligent Driver Model (IDM) as a simple car-following model with descriptive parameters.

Mechatronic Layer

The mechatronic level must be designed for cooperative vehicles, which is done by `Stiller et al.` (cf. [329] p. 215) They argue that the potential for improving traffic efficiency and safety beyond the level reached by human drivers will come through long term automated cooperation among traffic participants. The authors give an overview of the hardware architecture (on page 217) including, on the lowest level, the vehicle itself with actuators and sensors, an active camera platform, and other sensors like GPS, radar and lidar. In addition, the platform and vehicle controller are implemented with embedded systems, and on top of this are an image and knowledge processing unit and a radio unit for communication. This hardware architecture has been used 'in three Audi Q7 and a Volkswagen Passat', as well as 'a Smart Roadster and a Volkswagen Touareg', in order for them to act as six fully autonomous vehicles.

On top of this hardware architecture, `Stiller et al.` proposed a functional system for cooperative cognitive automobiles based on a model of human behavior (p. 215), which places skill-based behavior, which depends on environment information, on the bottom, rule-based behavior in the middle and knowledge-based behavior on top.

The system architecture presented in Figure 2.6 (p. 216) includes knowledge representation, as well as the other representations of the physical vehicle layer with its abilities, tactical and controlling dynamic system layer, and a global situation analysis and behavior generation. The stand-alone models of perception and behavior generation use information obtained in cooperation with other vehicles. The cooperation of vehicles consists mainly of cooperative perception, for instance in the detection of rear traffic, views into blind spots, views around curves, enhanced range and plausibility validation in overlapping fields of views. From artificial intelligence rule-based and knowledge-based cognition methods were modified. Thus, measurement uncertainties or contradictory rules can be handled with confidence measures: rule-based a posteriori probabilistic reasoning with Marcov models is used or representation with rules in a logic ontology. The idea to give the perception process more accurate information with the help of probabilities is good. Where robust information is gained by the fusion of data from different sensors and used, the success guarantors for autonomous driving are object tracking with Extended Kalman Filters or particle filters and model based object detections. Future research includes cooperative passing and emergency brake maneuvers using the success of cooperative city cars described in `Baber et al.` [12].

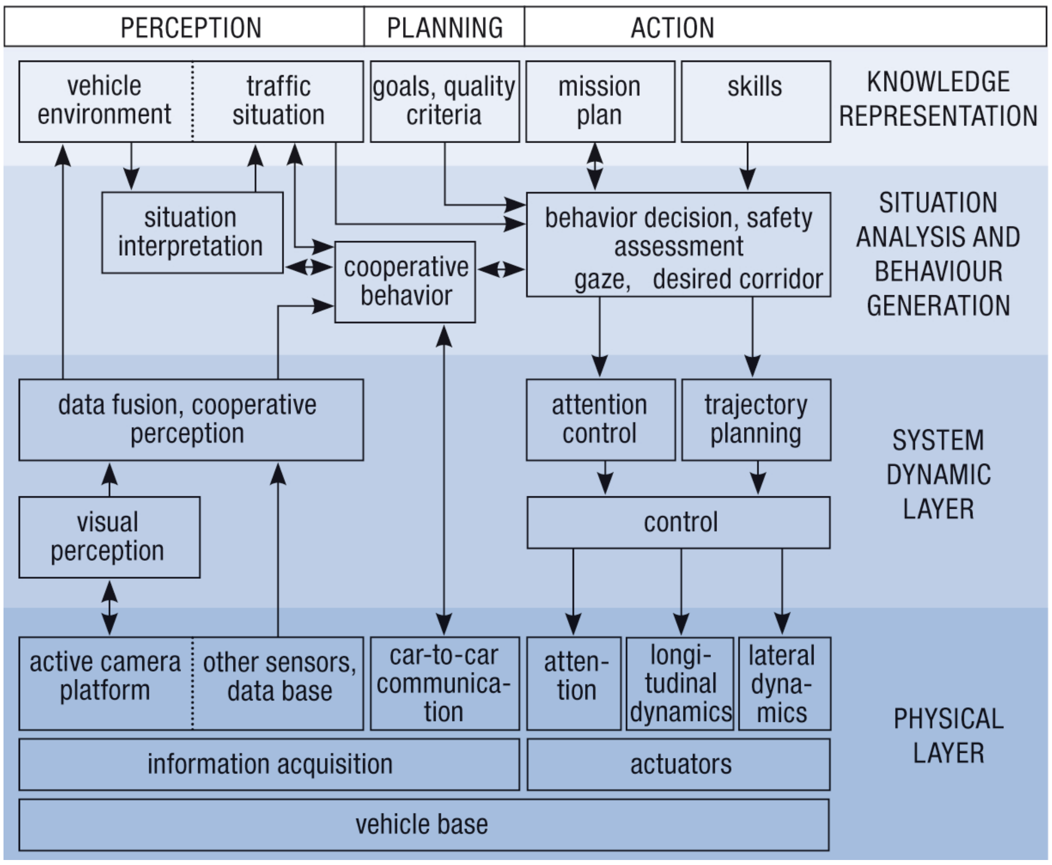


Figure 2.6: Cognitive Architecture (cf. `Stiller et al.` [329] p. 216.)

Compared to the focus of this thesis on dynamic group formation, the paper of `Stiller et al.` [329] gives a good underlying architecture which is implemented in real vehicles, but cooperative behavior is still to be implemented.

2.2.5 Traffic Simulation Systems

Traffic simulation is the mathematical modeling of traffic and transportation systems (e.g., intersections, arterial routes, downtown grid systems) through the application of computer software to help plan, design and operate urban traffic systems in a better way. Simulation produces visual demonstrations of present and future scenarios of transportation networks and is used for experimental studies of complex models. These are classified according to whether time, state and space are discrete or continuous.

The general notion of simulation is described by `Klügl` [207] and illustrated in Figure 2.7: a model is used for experiments, where a model is an image of the real world. This model can be considered as a system and is used to obtain information about another system by the user. A system is a set of objects with a structure, where any regular form of interaction may be taken as a structure.

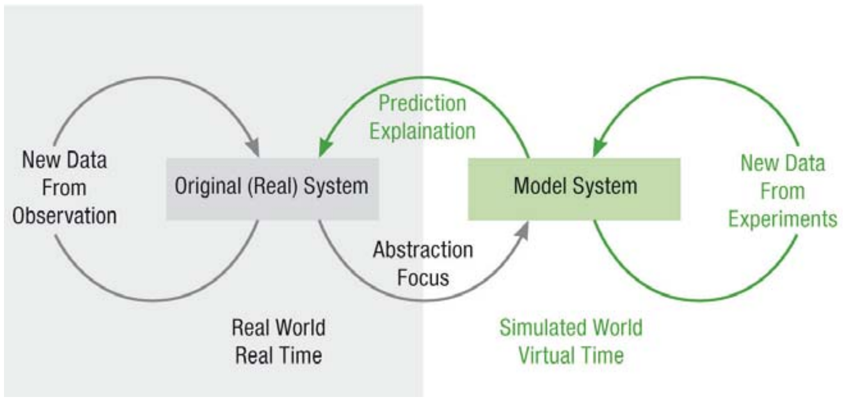


Figure 2.7: Relations between Real World and Simulation Model (cf. [207] p. 203).

Vehicular traffic systems are of global interest and growing concern. Modeling arbitrarily complex traffic systems is a hard problem and therefore abstracting the real urban traffic world in a microsimulation as a case study is necessary.

For understanding a simulation, the underlying concept of system state describes the evolution of the simulation over time. System state can be either discrete or continuous and is a set of variables that contains information about the environment [325].

There are many interactive traffic simulation systems publicly available [21], either commercially or open source, which model the macro-, micro- and/or mesoscopic perspectives. The focus here is on microscopic simulation.

First, the *commercial* traffic simulator AIMSUN is described. There are similar traffic simulators like PARAMICS[118] or VISSIM [281] [232], which offer all necessary functionalities (graphical presentation,modeling tool, automatic data collection and evaluation) and a variety of additions for emission or pedestrian models and interfaces. The problem of commercial products is that the underlying models are generally not sufficiently documented or accessible for developers. A more detailed survey of microscopic and macroscopic simulators is presented in `Ratrout` [289] and `Kotushevski` [211], where six different simulation softwares are investigated, and a functional evaluation of out of these above mentioned, three commercial traffic simulators is done in `Hidas` [167].

Second, the *open source* simulator SUMO (Simulation of Urban MObility) [215] is described in detail. MovSim [348] [386](based on Treibers Microsimulation of road traffic [242]) and MATSim [15, 16] are different open source projects. SUMO and Treibers Microsimulation of road traffic provide an important feature for free software packages: their source codes are freely available for download and for developer use. SUMO is actually an open source project that is being developed by two different institutions, in contrast to Treiber's Microsimulation MovSim which is a personal software project whose source code is available.

AIMSUN

AIMSUN is a simulation package of Transport Simulation Systems (TSS) that integrates all three types of transport perspectives: static traffic assignment tools, a mesoscopic simulator, and a micro simulator. The microscopic simulation is used to simulate small traffic scenarios. AIMSUN uses both models of`Gipps`, the car following model [132] and the lane changing model [133], to simulate the behavior of vehicles. The mesoscopic simulation perspective makes it possible simulate big traffic scenarios and both Gipps models are optimized for using less computational power. A traffic scenario can be generated automatically with a Geographic Information System (GIS) file[1]. A graphic tool is offered by AIMSUN to model different traffic scenarios. A simulated traffic scenario can be visualized with 2D or 3D animation and is saved in an AIMSUN or free choice database. External application can access the traffic objects with programming interfaces in Python or C language provided by AIMSUN. AIMSUN runs on Windows, Linux and MAC operating systems and is capable of communicating with Linux or MAC applications.

SUMO

Simulation of Urban MObility (SUMO) is an open source, highly portable, microscopic road traffic simulation package designed to handle large road networks [215] and implemented in C++. SUMO is documented, but doesn't facilitate API or plugin architecture. The simulation provides multi-lane streets with lane changing, including different right-of-way rules and traffic lights. Thus, in SUMO, modeling entire traffic networks is possible. The Krauss car following model [216] is used. SUMO provides a fast openGL graphical user interface and can manage networks with 10.000 edges (streets) at a fast execution speed. For network import VISUM, VISSIM, Shapefiles, OSM, RoboCup, MATSim, openDRIVE and XML-descriptions are supported and missing values are determined via heuristics. For routing microscopic routes are used, which means each vehicle has its own route and different dynamic user assignment algorithms are enabled. The simulation is continuous in space with time-discrete vehicle movement providing different vehicle types. Using the extension Traffic Control Interface (TraCI), traffic objects can be managed during simulation. In this case SUMO runs as a server and access is through TCP and/or binary protocol, where the client sets the simulation time steps. TraCI4J helps to serialize messages with Java support. Additionally, SUMO is connected with OMNeT++ to simulate communication.

2.3 Multi-Agent Systems

Multi-agent systems [382] provide new developments to software engineering. Complex systems are designed and modeled with high-level abstractions. The

[1] more information available at `www.gis.com`

process starts with requirements, then analysis, design architecture, implementation and, finally,testing and, therefore, is suited for this work for modeling complex traffic networks with agent entities.

The scope of multi-agent systems is very large and includes economics, philosophy, logic, ecology, and the social sciences. It concerns multi-agent environments as well, such as virtual training [287, 336], interactive entertainment [159], internet-based information integration, robots (space exploration rovers, RoboCup soccer [204], and robotic space missions), and sensor networks (weather tracking radars), etc.

Since its initial recognition in about 1990, several studies have shown a lot of attention to autonomous entities and confirmed significant results in computer science. The fields of distributed systems and object-oriented programming have a fertilizing influence on agent research. Therefore, `Weiss` ([373] p. 35f.) proposes three distinctions between the traditional view of an object and the view of an agent:

- On request from another agent, agents decide whether or not to perform an action. This illustrates the stronger notion of autonomy then objects;
- agents are capable of flexible (reactive, pro-active, social) behavior, whereas the standard object model has nothing to say about such types of behavior;
- each agent is assumed to have at least one thread of control, thus, a multi-agent system is inherently multi-threaded.

A MAS consists of the multiple computing elements called agents (for more detail refer to Section 2.3.1), which interact in a domain and are characterized by both making their own decisions according to the objectives they have to follow, and taking social responsibility for achieving cooperation, coordination and negotiation.

An agent in its environment and the top-level view of an agent are specified in Figure 2.8, which is adapted from `Weiss` [373]. The autonomous agent receives sensory input from the environment and the action output generated by the agent in turn affects the environment. The interaction illustrated with arrows is usually non-terminating and, thus, ongoing.

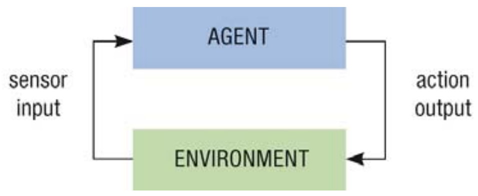


Figure 2.8: An agent in its environment, adapted from `Weiss` [373].

In most complex domains an agent does not have complete control over its environment, but, at best, partial control within its sphere of influence. That means that the same action performed twice by the agent in nearly similar conditions might result in different effects on the environment. Actions, therefore, have preconditions associated with them with possible applications to certain situations. The key problem of an agent is the reasoning about which action to perform next. The action must suit its design objectives best. These are defined by its architecture, described in more detail in Section 2.3.1.

Thus, agent systems are distinct from expert systems (cf. [185]), which are capable of solving problems or giving advice in knowledge-intense domains. Agent systems are situated in an environment, whereas expert systems do not interact with any environment and, therefore, inherently are disembodied. That means they do not get their information via sensors but through a middle man such as a user and, accordingly, they do not act on any environment, although they do give feedback or advice.

One of the difficulties for the designers of MAS is that executions and also negotiations are time consuming, depending on the level of detail. the basic concern of MAS research is the intelligent behavior of a collection of autonomous agents including the focus on understanding interactions between self-motivated, rational and autonomous agents. Even though the first agents to be developed act according to personal goals representing their own interests, there are cases where they should work for a common goal. So each of them might have different behavioral functions, but they have to have rational behavior as well, in an effort to maximize expected services. The design of agents should take the most optimized ways of achieving its own goals into account. This happens with respect to a definite set of shared resources, which are either limited or unlimited. The agents must either share a common resource, having the benefit of sharing an expensive resource, or they can derive benefit from sharing a set of common tasks.

First, regarding the task distribution problem, where sets of autonomous agents have a common goal to reach: they are usually asked to perform different tasks and agreements with other agents so that the common goal can be reached. Each agent mainly wants to satisfy the goals efficiently, but it also wants to minimize the costs by optimizing its actions. Negotiation protocols are used to make their cooperation possible, so they can fulfill their goals and avoid conflicts.

The agent systems in this work have especially the perspective of automatizing human behavior into intelligent traffic. Therefore, all aspects of traffic need to be modeled and five abstractions derived from `Boissier` [45] are applied to multi-agent systems:

- **Individual Agents:** Autonomy and situatedness
  Cognitive concepts: beliefs, desires/goals, intentions/plans
  Deliberation and decision: sense/reason/act as actions and perceptions interacting with the environment, reactive/proactive, processing internal events using the JASON Agent Programming Language
- **Multiagent Organization:** Social and organizational structures
  Groups, Missions and Roles: functionalities, activities, and responsibilities

Organizational rules: constraints on roles and their interactions called deontic relations, norms, deadlines, obligations, sanctions and rewards
Organizational structure: topology of interaction patterns and the control of activities using the organizational model for Multi-Agent Systems based on notions like roles, groups, and missions for programming multi-agent organizations (MOISE) framework

- **Environment:** sensing and acting in an environment, resources and services that Multi-agent Systems (MAS) can access and control, legacy and objects using the Common ARTifact infrastructure for AGents Open environments for programming environment artifacts (CArtAgO) platform

- **Interaction:** Speech Acts, Communication Languages, Interaction protocols using the JADE platform

- **Simulation:** combines the four elements mentioned above into one simulation using the MATI framework described in Chapter 4.

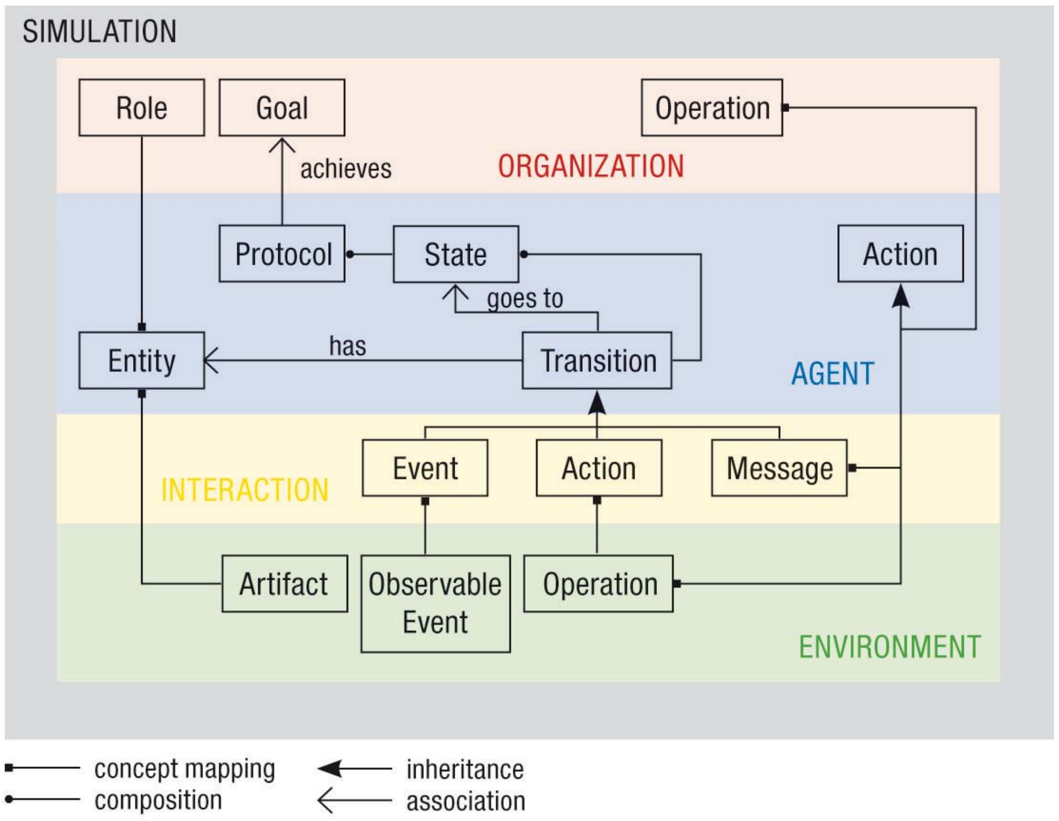


Figure 2.9: The Agent-Environment-Interaction-Organization concept (cf. `Zatelli and Hübner` [388]).

2.3.1 Agents

The main point about agents is that they are autonomous and capable of independent action, illustrated in Figure 2.10 as a vehicle perceiving its environment, adapted from `Wooldridge` [382]. The environment gives the agent feedback through the sensors and/or communicator. With this perception the agent reasons about the next action to take in the environment. The chosen action is executed in the environment through the effectors, in this case the wheels of the vehicle.

Agents extend other forms of modular and object-oriented software by exhibiting degrees of autonomy and pro-activeness. The agents' autonomy refers to the notion of entities that are designed to have a certain control over their individual actions which is independent from other systems or human intervention. Autonomous agents act on a set of predefined or dynamically generated goals. Small software entities are considered to be agents that work together to form Multi-agent Systems (MAS). Classic programs, on the other hand, are considered to be purely reactive artifacts, that is, software applications with a limited lifespan that are invoked by a triggering mechanism, execute a task and wait for further calls.

Intelligent Agents

`Weiss` [373] (p. 32) defines an intelligent agent as

> capable of flexible autonomous action in order to meet its design objectives

Flexibility intelligent agents exhibit three types of behavior: reactive, proactive and social `Wooldridge` [381].

Reactivity: A program can execute blindly if its environment is guaranteed to be fixed. In the real world, most environments are dynamic and not designed to be static. But software is hard to build for dynamic domains because a program must take the possibility of failure into account and check whether it is worth executing the tasks in the environment. A reactive system maintains ongoing interaction with its environment and responds to changes in it within a time frame in which the change matters.

Pro-activity: With a stimulus and resulting response rules it is simple to react in the environment, but generally humans want agents to do things (which can be automatized) for themselves. Therefore, goal-directed behavior is desirable. The idea of pro-activeness is to generate and attempt to achieve goals which are not solely driven by events but come from the agent's own initiative. Also, opportunities need to be recognized.

Social ability: The real world is a multi-agent environment. Without taking others into account we are very limited and some goals can only be achieved by interacting with others. This also applies to computer environments such as the Internet.

Nowadays, with the knowledge of these intelligent agents, there is a new trend to include decision making and interaction of multi-agent environments to widen the 'cooperation-oriented view of MAS' and find useful environments for their application, like the complex environment of urban traffic.

Typology of Agents

Nwana and Ndumu [263] classify agents in total into five categories:

Firstly, by their mobility, i.e., by their ability to move around some network, which yields the classes of static or mobile agents.

Secondly, either deliberative or reactive agents. Deliberative agents derive from the paradigm that agents possess an internal symbolic reasoning model, and plan and negotiate with other agents in order to achieve their goals. Reactive agents originate from robotic research by Brooks [53] and act using a stimulus-response behavior, which reacts to the present state of the environment in which they are embedded [100].

Thirdly, by the minimal attributes they should exhibit: autonomy, learning and cooperation. Autonomy refers to the key element of pro-activeness and the principle that agents can operate without the need for human guidance. Learning means that, as agents react and/or interact with their external environment and others, with time, their performance improves. Cooperation with other agents is important and the reason for having multiple agents; communication supports this.

Fourthly, by their roles, e.g., Internet information gathering agents, which help to manage the vast amount of information in wide area networks of the world wide web. This class of agent is referred to as an information or Internet agent, which can be static or mobile and deliberative or reactive.

Fifthly, by the category of hybrid agents. In a single agent, they can combine two or more agent philosophies.

In essence, agents exist in a truly multi-dimensional space and six types of agents can be distinguished: collaborative, interface, mobile, information/Internet, reactive, and hybrid. Some applications combine agents from two or more of these categories. They are then referred to as heterogeneous agent systems.

The first three characteristics are used in Figure 2.11 to derive four types of agent to include in this typology: collaboration agents, collaboration learning agents, interface agents and truly smart agents. This research deals with cooperative and autonomous agents according to this typology, that is, with collaboration agents. The field of learning is only slightly touched by the route choice of agents in the network, but it is not the focus of further investigations.

These distinctions are not definitive, but give a good overview of the interdependency of autonomous, learning and cooperation agents. Collaborative agents have more emphasis on cooperation and autonomy than on learning, but that does not imply that collaborative agents never learn. Only what falls within the 'circles' is considered to be an agent. Although expert systems are largely autonomous when giving advice to a domain, typically they do not cooperate or learn.

This research makes use of the agents and artifacts (A&A) modeling paradigm [173], which introduces agents, artifacts and workspaces as first-class abstractions.

In the context of agents and Multi-Agent Systems (MAS), the artifact abstraction models [2] construct parts to act as a working environment for agents. Therefore, resources and tools are constructed as artifacts, which are shared and used by agents to support their activities, both individual and cooperative. One example artifact can be a traffic light, which changes its signal phases and is observed and acted upon by the vehicle agents.

The A&A is investigated as a general-purpose paradigm for designing and programming software systems, in particular concurrent and distributed ones. In the context of complex systems it adds changing artifacts to the basic agent-based model for multi-agent based simulations (MABS).

Agent Architectures

Intelligent agents can be formulated into three main abstract architectures [373]:

1. the purely *reactive agents*, which simply respond directly to their environment
2. *perception and action systems* with the function of observing its environment and the action of representing the agent's decision making process
3. *agents with state*, where the agent starts in some initial internal state, observes its future environment state, generates percepts and updates its next function. This way it can foresee the future steps and its changes and, therefore, can choose an action to perform appropriately in the environment, like the computer in a chess game.

The first architecture would represent normal state-of-the-art vehicles well, whereas the second one is well-suited for autonomous vehicles due to their own reasoning about their observations and executions. The third architecture is favored for discrete models like the cellular automata model by `Nagel and Schreckenberg` [259], mentioned in the microscopic car following models in subsection 2.2.4.

Constructing agents requires an architecture which is specified in four classes of agents [374] [382]:

- *logic-based agents* describe decisions about the next action in a logical notation
- *reactive agents* describe that decision-making underlies the action in a certain situation
- *belief-desire-intention agents* describe the decision-making through manipulation of data structures which represent the beliefs, desires, and intentions of the agent

[2]details described on the website `https://apice.unibo.it/xwiki/bin/view/Themes/`

- *layered architectures* describe the reasoning about decision- making in software layers representing the environment on different abstraction levels.

Belief-desire-intention (BDI) agents and layered architectures are the most promising for autonomous vehicle agents and their task of group formation.

BDI agents have their roots in practical reasoning from philosophical tradition [52], which incorporates the process of deliberation about which goals to achieve and the process of mean-ends reasoning as to how the action plan furthers the goals [127] [285].

Layered architectures [102] [257] can have two types of control flow: vertical layering, where the sensory input goes through all layers at least once to the action output, and horizontal layering, where each software layer is directly connected to the sensory input and action output. The InteRRaP architecture [254] and another 3T intelligent control [46] are examples of layered architecture.

Planning

The agent can have two types of goals: its individual and the joint goal. The individual goal of each vehicle is to drive at its desired speed. The agent chooses its lane so that its utility is maximized according to its route preference.

A general utility function and the cost estimation function are defined by `Müller` ([253] p. 90):

Definition 2.1 (General Utility Function) *Given a finite set $\mathcal{P}$ of plans, a utility function for plans is a function $u : \mathcal{P} \mapsto \mathbb{R}$, mapping plans into real numbers. The utility of a plan $p \in \mathcal{P}$ is computed as $u(p) = w(p) - c(p)$, where $w : \mathcal{P} \mapsto \mathbb{R}$ is a worth function, and $c : \mathcal{P} \mapsto \mathbb{R}$ is a cost function for plans.*

The worth of a plan is equal to the worth of a goal achieved by successfully executing the plan. Regardless of the definition of the function w, alternative plans which achieve the same goal $w(p_i) = w(p_j)$ are chosen. Therefore, the focus can be on comparing the costs of plans. A finite set $\mathcal{P}$ and a cost function c are given with a formalized plan selection *select* : $2^{\mathcal{P}} \mapsto \mathcal{P}$, which returns a plan p with

$$c(p) = \min_{p_i \in \mathcal{P}} c(p_i).$$

In case plans are sequences of atomic actions $a_1, ..., a_n$, the costs are allocated straightforwardly: starting from a cost function $\hat{c}$ for atomic actions, the cost of plan p is calculated by

$$c(p) \sum_{a_i \in p} \hat{c}(a_i).$$

. However, the plan can be executed depending on a set of predicates $\alpha_1, ..., \alpha_n$ defined in the knowledge base hierarchy and with two language control expressions: conditional plans use if-then-else constructs and iterations can be expressed with while-do constructs (refer to `Müller` [253] p. 85). Therefore, the selection of plans is done during planning time and costs have to be estimated. `Müller` assumes the existence of a cost function for primitive plan steps. This thesis gives a cost function for traffic in Section 5.3.1. A probabilistic model for a priori costs is needed for the evaluation of conditionals and iterations in order to compute probabilities of conditions or to predict statistically expected values for how often an iteration is carried out. Those models are nontrivial and require domain knowledge from traffic. The discussion of statistical domain models exceeds the scope of this thesis because they are not crucial for autonomous vehicle groups and are a separate area of research.

The following cost estimation function provides a general method for choosing among different applicable plans.

Definition 2.2 (Cost Estimation Function($\mathcal{L}_{\prime}$)) *Let $\hat{c}$ be a cost function for primitive plan steps. A cost estimation function for $\mathcal{L}_{\prime}$-plans is a function $c_0 : \mathcal{L}_{\prime} \mapsto \mathbb{R}$ with:*

- $c_0(p) = \hat{c}(p)$ *for primitive plan steps* p.
- $c_0(p_1, ..., p_n) = \sum_{i=1}^{n} c_0(p_i)$ for $p_i \in \mathcal{L}_{\prime}$ *(sequential composition).*
- $c_0(p_1; ...; p_n) = \max_{i=1...n} c_0(p_i)$ for $p_i \in \mathcal{L}_{\prime}$ *(disjunctive composition).*
- $c_0(\textbf{if } e \textbf{ then } p_1 \textbf{ else } p_2 \textbf{ fi}) = Pr(e) \cdot c_0(p_1 + (1 - Pr(e)) \cdot c_0(p_2)$ *for* $p_1, p_2 \in \mathcal{L}_{\prime}$.
- $c_0(\textbf{while } e \textbf{ do } p \textbf{ od}) = E(e, p) \cdot c_0(p)$ for $p \in \mathcal{L}_{\prime}$.

Pr is a probability distribution over conditions and $Pr(e)$ denotes the probability of e being true in a certain world state. $E(e, p)$ denotes the statistically expected value for the number of iterations of p required to reach a world state in which e does not hold.

Due to the inductive definition of the plan structure, the properties of two plans $p_1, p_2 \in \mathcal{L}_{\prime}$ are defined such that p_1 is syntactically a sub-plan of p_2, which means a plan step contained in p_1 is also included in p_2, denoted by $p_1 \subseteq p_2$ results in $c_0(p_1) < c_0(p_2)$. The costs of the first plan are smaller than the second plan, because the first plan is included in the second plan.

Agent Programming Languages

`Wooldrigde and Jennings` [380] summarize agents, architectures and languages. The need for agent software tools was identified by `Gasser et al.` [122] and this section will name only some known agent programming languages. The first to be introduced was agent-oriented programming by `Shoham` [318] with the prototype implementation of AGENT0, where the agent is specified through capabilities, initial beliefs, commitments and rules. The Concurrent MetateM [114] is based on logical formulas.

Some of the known languages are PLACA [342], MAIL [156], AgentSpeak(L) [286], 3APL [168], dMARS [87], and TELESCRIPT [337].

An overview of agent-oriented languages is given in `Bordini et al.`[48]. Agent programming languages are not easy to distinguish from the simulation software, Some are evaluated with the agent simulation platforms in Section 2.3.5.

2.3.2 Environment

Agent interaction and coordination is done on the platform of the environment, therefore the protocols are described in the following.

According to Weiss [373], coordination is a characteristic of the system when agents perform their actions in the shared environment. Cooperation, illustrated in Figure 2.12, is coordination among good-natured agents. Negotiation is the kit to coordination between competitive or self-interested agents. For a successful cooperation an agent needs to include the other agents into its world model, including a model of future interactions. Therefore, they need to plan what can be executed by distributed or centralized planning.

Limited resources in an environment trigger agents to coordinate their activities with other agents in order to fulfill their own interests or satisfy group goals. Due to dependencies of actions and global constraints, the coordination of multiple agents is important for achieving the system goals for which no single agent has sufficient competence, information or resources.

The control of the environment may be centralized and static like today's traffic management, but distributing control and data can also produce coordinated systems. Distributed control is when the agents have the autonomy to choose their actions and generate their own goals. The necessity for distributed control arises because the knowledge of the system needs to be distributed and available for all participants. Usually, the agent has only a partial and individual perspective on the global system, although it might be beneficial for distributed control to enhance the individual situation leading to better coherent system performance.

For the system design, activities include defining the goal(s) for both perspectives (individual and global), assigning particular regions to specific agents, controlling decisions about what to explore and ensuring good documentation.

The key agent structures for coordinating global and local problems are *commitments*, providing structure for predictable interactions, and *conventions*, providing the degree of mutual support described in detail in Jennings [191] (p. 7f). Based on ideas of Durfee et al. [95] Jennings [191] (p. 11f) extends the three major points for successful coordination, quoted in the following items:

- there must be *structures* which enable the agents to interact in predictable ways;
- there must be *flexibility* so that agents can operate in dynamic environments and can cope with their inherently partial and imprecise viewpoint of the community; and
- the agents must have *sufficient knowledge and reasoning capabilities* to exploit the available structure and the flexibility.
- Additionally, structures to provide *mutual support* to the cooperating agents are necessary.

Thus, to support the last item, cooperation needs to be defined between the environment and the multi-agent system and between the agents and their

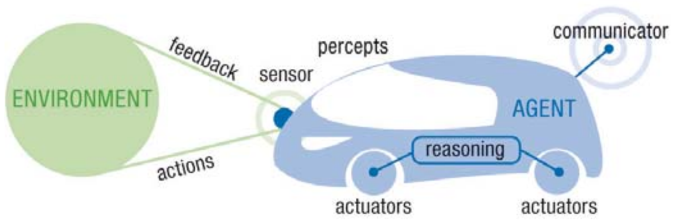


Figure 2.10: A Vehicle as an Agent (adapted from `Wooldridge` [382]).

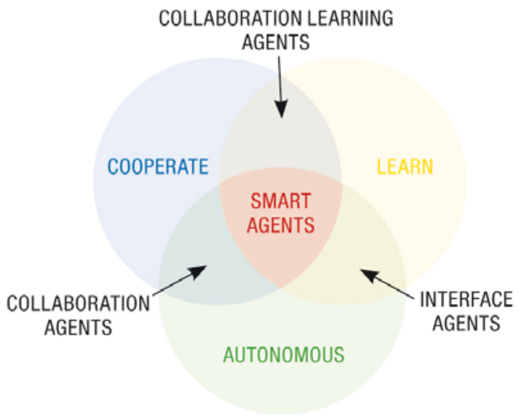


Figure 2.11: Typology of Agents (cf. `Nwana and Ndumu` [263] p.6).

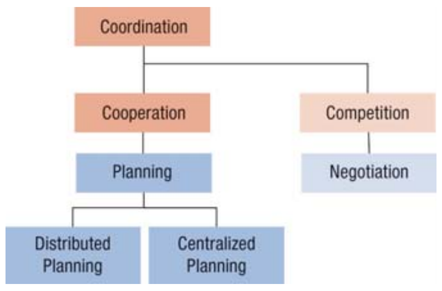


Figure 2.12: Coordination taxonomy (adapted from `Weiss` [373] p. 83).

interaction with each other and the environment. Cooperation protocols usually decompose and distribute tasks Weiss [373] (p. 99). Mechanisms are:

- Market mechanisms with matchmaking strategies (like auctions) by general agreement or mutual selection
- contract net to announce, bid and award cycles
- multi-agent planning where specialized planning agents are responsible for assigning tasks
- organizational structures to assign fixed responsibilities to particular tasks

This research focuses on Contract Net protocols (CNP) as described in Weiss[373] (p. 100f.) and the organizational structure of a social system, elaborated in Section 5.3.1.

According to Klügl [207] (p. 207), the dimensions of characterizing on the environmental level including the meta level are mapped to urban traffic:

- topology of space: urban networks
- role and environmental richness: street networks with focus on vehicular traffic
- objective of simulation study: benefits of AVGF
- level of abstraction, empirical embedding: street networks with their infrastructure without buildings or pedestrians
- extent of stochasticity: microscopic modeling and realistic data from urban network

2.3.3 Interaction

According to Weiss [373], agents interact in order to achieve better goals for themselves or for the environment where they exist. Communication is the key for agents to coordinate their actions and behavior, resulting in more coherent systems. Three aspects are included in communication: the syntax, which is the structure of symbols; semantics, which is the denotation of the symbols; and pragmatics, which is the interpretation of the symbols.

Russell and Norwig ([300] p. 904) describe communication as follows:

> Communication is the intentional information exchange brought about by the production and perception of signs drawn from a shared system of conventional signs. [...] In a partially observable world, communication can help agents be successful because they can learn information that is observed or inferred by others.

Natural language understanding ([300] p. 935) is complex and machine translation systems implement a range of techniques based on frequencies of messages: ranging from full syntactic to semantic analysis to statistical techniques. Speech recognition is also primarily based on statistical principles and is popular and useful. Currently, statistical models are most popular and successful. Together, machine translation and speech recognition are two big successes of natural language technology.

`Singh` [321] defines the dimensions of meanings that are associated with communication.

- descriptive vs. prescriptive: descriptions are necessary for human comprehension, but difficult to mimic for agents. Most agent communication is designed to exchange information about activities or behavior.
- personal vs. conventional: usually, everyone has their own interpretation based on experience for a message, but this might differ from the generally accepted understanding. Agent systems should opt for conventional meanings because they are typically open systems and new agents can enter at any time.
- subjective vs. objective: the direct effect on the environment is objective because it can be observed by everyone, but internal interpretation is subjective.
- perspective of speaker vs. hearer vs. society: this expresses the different viewpoints of speaker, hearer or other observers.
- semantics vs. pragmatics: pragmatics affect the use of communication by the communicators and respects the environment as well as the individual considerations, which are excluded by syntax or semantics.
- contextuality: messages cannot be understood in isolation, therefore this includes the individual, the environment and the history of messages and actions.
- coverage: can be seen technically or as the expressiveness of a language.
- identity: the meaning of a message depends on the individual and its roles and specifications.
- cardinality: the different effect of a private or a public message.

Coordination is realized by communication, that is, information exchange (cf. [169] p. 19). Thus, communication is a necessary precondition for the formation of coordination, such as groups.

According to `Teufel et al.` [341], there is a hierarchical relationship of three components with communication as the base, coordination in the middle, and cooperation at the top:

- communication: is the understanding and agreement of several parties among each other.
- coordination: identifies the communication which is necessary for the reconciling of task-oriented functions that are executed in the context of group work.
- cooperation: is denoted by the communication which is required for coordination and agreement of common goals.

Multi-Agent Communication

Communication is important for agents to coordinate themselves and it is one property of being social. Different capabilities which agents need for communication are: to receive and send messages through a fully functioning communication network, and to participate either passively, actively or both in a dialog in order to assume the function of a master, slave, or peer respectively. The two basic message types are assertion and queries, further discussed in [373].

Communication can be defined on different levels, but it should be on the lowest level so that the least capable agent can take part in it. The lowest level is described by the interconnection method, the middle level by the information format or syntax and the top level by the meaning or semantics of the information specified in communication protocols. Those protocols can be binary, with a single sender and a single receiver, or broadcast, with an n-ary protocol involving a single sender but multiple receivers, also called multi-cast. The data structure is defined in a protocol containing the sender, receiver(s), the language of the protocol, encoding or decoding functions and actions to be taken by the receiver (compare to [373] p.86f.)

Speech Acts

Computational agents use spoken human communication as a model defined by speech acts by the philosopher John Searle [314]. Speech acts can be seen as actions in human communication such as requests, suggestions, commitments, and replies. Thus, speech act theory helps to formalize the types of messages which are used in computer science [353], especially multi-agent systems (MAS) [63, 320].

There are at least two standardizations of speech act labeled messaging. KQML [220] and the Foundation for Intelligent Physical Agents (FIPA), established in 1996, have given mentalist semantics to agents.

Thus, the language used for MAS is the Agent Communication Language (ACL) [221].

Later approaches [70, 115, 194, 195] gave social semantics to ACL.

The Knowledge Interchange Format (KIF) [126] is a logic language and a proposed standard for describing things within expert systems, databases, intelligent agents etc. KIF provides encoding of knowledge.

Ontologies describe domain knowledge of objects, concepts and relationships; the area of interest in this thesis is traffic [39].

The models discussed are the most notable, but there are also other means by which computational agents can interact, communicate and be interconnected; compare `Weiss` [373].

2.3.4 Organization

In a multi-agent system (MAS), the organization (of the traffic system) is useful for improving efficiency since the organization constrains the agents' behaviors towards social intentions, that is, their global common purpose. Without some degree of organization, the agents' autonomy may cause the system to lose global congruence (cp. [174] p. 118).

The organization models are divided into two perspectives: agent [153] or organization centered view [101]. In the former, the agents are seen as the engine for the formation of the organization, in the latter the organization is defined by the designer and exists a priori. Another classification of organizational models has been proposed by `Hübner et al.` [174] with the focus on:

- society's global plans or tasks: functional specifications
- society's roles: structural specifications.

Organization are categorized by four notions by `Boissier` [45] as follows:

- **organizations are supra-individual phenomena:** Organizations are structured, patterned systems of activity, knowledge, culture, memory, history, and capabilities that are distinct from any single agent [123].
- **Definition by the designer, or by actors, to achieve a purpose:** A decision and communication schema, which is applied to a set of actors, that together fulfill a set of tasks, in order to satisfy goals while guaranteeing a globally coherent state [239].
- **Pattern of predefined cooperation:** An organization is characterized by a division of tasks, a distribution of roles, authority systems, communication systems, and contribution-retribution systems [40].
- **Pattern of emergent cooperation:** An arrangement of relationships between components, which results in an entity, a system, that has unknown skills at the level of the individuals [252].

Organization infrastructures are e.g., Madkit [148], Karma [260], Ameli [98], S-Moise+ [172], MoiseInst and SYNAI [124], and cooperation is a focus of `Khamis et al`.

Organizations in MAS are defined as supra-agent patterns of emergent or (pre)defined cooperation with a purpose that can be specified by the designer or by the agents themselves. Patterns of emergent or potential cooperation are called organization entity, institution, social relations or commitments. Whereas patterns of (pre)defined cooperation are called organization specification, structure or norms.

The cooperation and coordination protocols are mentioned in the environment section 2.3.2.

2.3.5 Multi-agent-based Simulation

The general information of why a simulation is used and the general notions which are described in traffic simulation section 2.2.5 also apply for agent simulation.

For defining agent-based simulation, which is equivalent to multi-agent simulation, the basics of a multi-agent model and an agent are defined as ([207] p. 205):

> A multi-agent model is a representation of an original system based on the conceptualization as the multi-agent system. The active entities in the simulation are mapped to 'agents'. An agent is an entity that is situated in an environment and is able to perform autonomous actions in this environment. It determines its behavior based on its own mostly local perspective and has only restricted/local manipulation capabilities.

Based on `Klügl` [207], the objectives for agent-based simulation are empirical understanding simulated in case studies, normative understanding, qualitative insight and theory generation and models as tools. Using agent-based simulation enables the new agent paradigm for more appropriate modeling for socio-technical systems such as traffic, makes it possible to treat systems as emergent phenomena and variable structure models, and provides an intuitive way of modeling to facilitate communication and visualization.

Multi-Agent Software has different ideas of autonomy like agent autonomy within the model versus autonomy in relation to the modeler, explicit macro- or micro-level observations, distinct treatment of simulated time and environment, and established guaranteed relation to a reference system which constrains the agents' behavior.The responsible modeler is in charge. Some of the software used for this thesis is described in detail; other propriety software is, for example MADKIT [149]and JACK [171].

Jade

JADE (Java Agent Development Environment) [36] is a software framework by Tilab[3] with the objective of implementing standard agent-based applications. JADE is implemented in the Java programming language and complies with the specifications of the organization 'The Foundation for Intelligent Physical Agents (FIPA)'.

The JADE framework can be viewed from two perspectives: runtime and agent development.

First, from the perspective of system runtime, JADE offers several services for agent-based applications. The multi-agent system constructed in JADE is a distributed system and is able to divide itself among different hosts. On every host an agent container exists, which administrates the agents and is part of the system. The agent container exists in its own Java virtual machine (JVM) and adds and saves a newly generated agent and deletes the agent when requested by the system. JADE offers a mobility function for the agents to move from one container to another. The communication between agents is realized with the standardized agent communication language (ACL) for exchanging messages. For observing the multi-agent systems, different tools like 'Remote Manager Agent' and 'Sniffer Agent' are available.

[3]The latest version of JADE is JADE 4.3.2 released on 03/28/2014 `http://jade.tilab.com/`

Second, from the perspective of agent development, JADE offers extensive libraries to support implementing agents and debugging of FIPA-conform agents. For developing a new agent, the abstract Java agent class is extended. Every agent is implemented as a thread, but, for cooperative behavior, multi-threading is possible in order to execute parallel tasks. This predefined behavior simplifies the construction of agent behavior. JADE supports asynchronous communication.

JADE provides no internal agent structures and offers no concepts for group oriented programming.

Jason

One agent programming language identified as suitable for autonomous vehicle group formation (AVGF) is Jason.

Jason [49] is an open source interpreter for an extended version of the AgentSpeak programming language. Jason implements the operational semantic of the language in Java and Prolog and offers a platform to develop multi-agent systems for the main operating systems Windows/Linux/MacOS. Jason was developed mainly by Jomi F. Hübner (Department of Automation and Systems Engineering, Federal University of Santa Catarina, Brazil) and Rafael H. Bordini (Instituto de Informatica, Universidade Federal do Rio Grande do Sul, Brazil).

Jason incorporates the BDI model and supports the development of environments (normally implemented in AgentSpeak) which are programmed in Java. Jason has an integrated development environment either in jEdit or as an Eclipse plug-in and includes a so-called 'Mind Inspector', with which, during runtime, agent behavior can be observed in detail. This isalso useful for debugging. General documentation is provided by the Jason book [49], also the homepage[4] has a tutorial and an overview with links and FAQ as well as publications and API documentation. An important insight is the reasoning cycle of Jason, which is inserted in Figure 2.13 from the original `Bordini et al.` [48].

Jason supports organizational agent aspects with the extension Moise [173], based on notions like roles, groups, and missions. `Hübner et al.` provide a tutorial with a soccer team. Jason uses the FIPA (IEEE Foundation for Intelligent Physical Agents, 2012) libraries for the communication between agents. The multi-agent system can be distributed with the help of Saci or JADE. The Multi-Agent Contest [32] is co-organized by the TU Clausthal.

2.3.6 Related Work (on MATI Framework)

This section describes the Agent Programming Toolkit (AplTk) and the Environment Interface Standard (EIS) as related work from `Behrens` [31]. The Environment Interface Standard (EIS) framework is used in the interpreters Goal and a former version of the programming language 2APL. The goal is to combine environments with agents no matter which interpreter is used,even if

[4] http://jason.sourceforge.net/

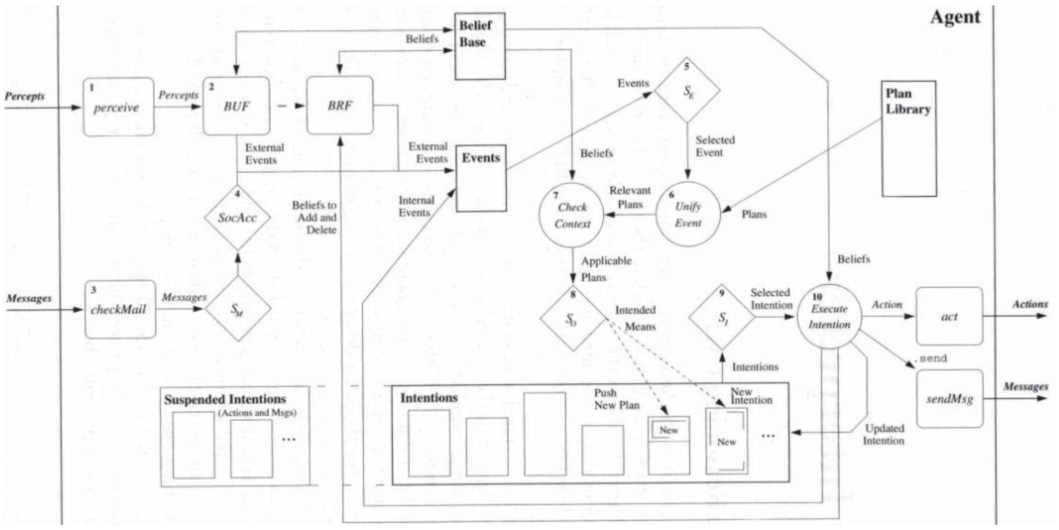


Figure 2.13: Reasoning Cycle Jason `Bordini et al.` ([48])

more than one are used. This results from using different environment simulation platforms together with agent simulation, which is presented in Chapter4.

Agent Programming Toolkit

AplTk is a universal *Agent Programming Languages Toolkit* for MAS. It consists of four main parts: the world. Based on the perceptions received, the agent will then select one or many actions to perform in the environment. In AplTk, communication between environments, interpreters and tools is done only through well-defined interfaces and the cardinality of these entities is many-to-many. Furthermore, AplTk possesses a tool and a core component. The tool interface ties in specialized tools for analyzing the output of generated modules (interpreters and environments,) which can be used for heterogeneous MAS. It can also be used with several different interpreters and environments. The core component integrates all the other components and acts as a scheduler where a scenario definition is assigned in form of an XML file. Using this definition, the core thread generates all components and executes the simulation.

Environment Interface Standard

The Environment Interface Standard (EIS) was designed to simplify the interconnection of agent platforms with environments. EIS was developed to decouple agents from their environment in MAS. Agents, seen as software structures to perceive and process data from the environment, are coupled with the interface layer where controllable entities (like vehicles) are connected. These are software structures which act to generate percepts and process agents' actions in the environment. The agent-entities relation as described in [31] gives an identifier to each agent and entity through the interface layer. Through this interface it is possible to exchange agents or environments and to compare them

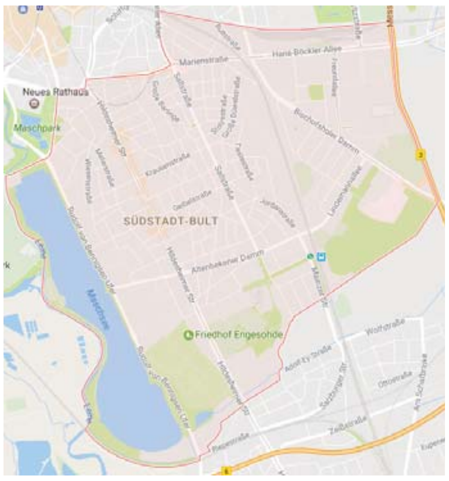


Figure 2.14: Representation of the Southern Part of Hanover, "Südstadt" by googlemaps.

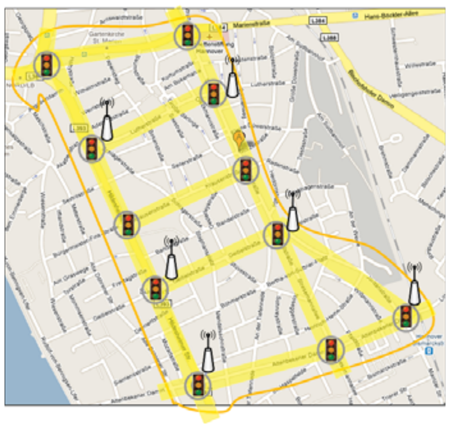


Figure 2.15: Scenario with traffic lights and Road Side Units by openstreetmap.

Figure 2.16: Southern Part of Hanover, Germany.

with tools. Thus, EIS promotes modularity and recyclability. It introduces percept-generators and action-processors on the environment side. The agents can control one or many controllable entities. Because EIS makes use of the *observer pattern*, listeners for the environment and agents have to be attached. Once the listeners are registered, perceptions and actions can be received.

2.4 Requirements for Vehicle Groups in Traffic

Based on the scenarios presented in the following subsection 2.4.1, requirements are derived in the next subsection 2.4.2.

2.4.1 Scenarios

A realistic traffic network of the southern part of Hanover, Germany is considered for this research, illustrated in Figure 2.16. The left representation, Figure 2.14, shows the district of the southern part of Hanover called 'Südstadt'. The right map in Figure 2.15 illustrates the scenario network used, with two parallel and five perpendicular streets. The streets are modeled with the traffic lights and Road Side Units (RSU) and considered to have the same priority (no distinction between main and minor streets).

In March 2009, one hour of empirical intra-urban commuter traffic data was collected in rush-hour traffic between 7:30 and 8:30 am. Traffic flows and traffic signal programs were parameterized accordingly as well as control programs being in operation. This realistic network scenario is prone to congestion. It issued for simulating the new idea of autonomous vehicle group formation

(AVGF). Thus, it provides the combination of centralized control with novel decentralized group decisions made by the autonomous vehicles.

Due to realistic data being available for this scenario, it is used for deriving three equivalent networks which differ in size. One extraction is made in order to consider only one arterial for a green wave scenario, here, only Hildesheimer Street (the left street) is used. Additionally, for research purposes, two artificial scenarios with the same traffic flow were constructed: a mini scenario with three lanes only and, for scalability, a grid scenario with ten intersections. Simulation will need to demonstrate the influences of AVGF in the different network scenarios.

2.4.2 Requirements

Following the presented scenarios, verifiable requirements for simulation of decentralized autonomous vehicle groups in urban traffic are defined as:

1. *Microscopic simulation environment:* Necessity of model and simulate urban traffic including dynamic traffic control as an open system. Use, enhance, develop or integrate the traffic simulation system such that defined strategies for autonomous vehicle groups can be assessed under real world scenarios as described above in Section 2.4.1. This implies real-world applications with real-world data. Thus, the correct description of dense traffic which comes from real-word Floating Car Data (FCD) data or detectors is important. Additionally, the environment as well as the vehicle agents should act in a cooperative manner, in order to achieve improvements from the individual as well as the global perspective.
2. *Modeling and simulation of autonomous vehicles:* necessity of vehicular communication and functions for self-driving properties such as availability of information and internal logic for reasoning about the environment. The environment as a map of the network is usually known to drivers as well as agents, but variable information is communicated to or observed by the agents within the sphere of influence. Communication between the agents should be possible, but it is liable to interference. Therefore, it is necessary to reduce the communication costs by using it as little as possible, but as much as needed. Since the agent paradigm is assumed to be most suitable for this approach, microscopic behavioral models for traffic should be implemented, which adopt to and perform in the dynamic urban traffic environment. The states of the agents should be represented in a continuous way instead of a discrete manner. Therefore, they are dynamic agents and move within the traffic environment as vehicles.
3. *Integration for coordinated actions of AVGF:* necessity of intelligent usage of dynamic vehicle groups. Definition of a set of acceptable vehicle grouping strategies that will run in urban traffic without changes to the road and roadside infrastructure. Regarding the organizational architecture without explicit stops to initiate vehicle grouping is called the 'non-locality' of group formation, which means that autonomous vehicles look ahead

and adapt their behavior to the traffic situation at their actual position as opposed to very regulated coordination booths. In this context, the main difference is centralized versus decentralized group algorithms. Interest groups are not evaluated in this work.

4. *Illustrate the economic benefit:* necessity for a quantification of achievable travel times. The thesis approach can be used to encourage the use of autonomous vehicle groups with benefits to both lead vehicle driver and to platoon member. The vehicle agents perform their tasks independent of each other, but they can coordinate themselves into groups in order to reduce travel and stop times through smaller gaps between the vehicles when driving in a group. Show how vehicle groups can lead to environmental, safety and congestion improvements. In order to analyze the group effects on traffic, the vehicle parameters in the system will be tested in homogeneous versus heterogeneous vehicle groups.

Autonomous vehicle groups are addressed in this thesis as a solution to organizational traffic problems. This research uses autonomous vehicle group formation as an instrument of differentiation.

You know more than you think you know, just as you know less than you want to know.
Oskar Wilde, (s. XIX)

Chapter 3

State of the Art

Autonomous Vehicle Group Formation is used to define and depict the tasks which need to be performed when executing a joint action, as well as the environment, which must be taken into regard during these tasks. Various methods have been developed to support the simulation of group processes in different contexts. In this thesis, five categories of dynamic group formation are distinguished, depending on their main area of application and origin.

First, the environment and traffic control are discussed in section 3.1. Second, the use of agents for modeling and simulation in traffic is described in section 3.2. Third, interactions including traffic communication are examined in section 3.3. Fourth, organizations and their traffic architectures are investigated. Most importantly, the fifth section 3.5 provides the investigations into group models.

3.1 Environments: Traffic Control

First, there are simulations for the environment including traffic control (TC), which are mainly used in urban planning for depicting various assumptions and characteristics of traffic management (TM). Traditionally, control is static, meaning in an off-line and traffic-independent fashion with a central architecture. All data is collected in a Traffic Management Center (TMC) on a high performance computer, which usually controls the traffic infrastructure from the top down, based on the historic data. Traffic modeling and simulation (MS) supports traffic regulation which is often steered by traffic lights (synonym: signals or controllers). Then, traffic flows are managed according to traffic patterns and based on buried detector data. As said by `Bazzan and Klügl` [29]: a priori determination of suitable signal plans according to traffic patterns at different times of the day (not to mention actual traffic loads) requires a lot of domain and network knowledge. Because traffic is a highly dynamic process, simple off-line or on-line optimization for synchronizing arterials alone cannot control various traffic flows during different times of the day. Due to increasing volumes of traffic, there is a demand for more flexible but robust approaches which should vary in their degree of sophistication and their scope. Nowadays, mobility challenges researchers to create new concepts with more flexibility and adaptability, and new equilibrium. Interdisciplinary research is necessary in complex systems dealing with highly dynamic traffic and transport problems. New technologies and methods like Dynamic Traffic Management (DTM), Intelligent Transportation Systems (ITS), C2X communications, improved data handling, human-like systems and simulations revolutionize the existing top-down and centralized traffic and transportation systems to create more dynamic, decentralized systems. These static, central and microscopic simulations, including urban simulations, provide the foundation for dynamic traffic management in the three other categories.

The aim of this thesis is to present a new combination of car-to-infrastructure communication (C2I, also known as C2X), Traffic Management (TM) and Intelligent Traffic Systems (ITS) with the agent perspective. Today, TM needs to deal with autonomy and communication on the part of vehicles in traffic. The more autonomous vehicles, equipped with sensors, that are in use today, need to be coordinated according to their behavior.

Intelligent Transportation Systems (ITS) [180] implies global control such as traffic signal control systems, variable message signs, or speed cameras for security CCTV systems, and more individual applications like car navigation, the integration of actual data and feedback from different sources such as parking guidance and information systems or weather information. Additionally, predictive techniques are contributing to faster and more accurate forecasts and learning effects for advance modeling with historical baseline data.

Agents and Multi-agent Systems (MAS) have been used in transportation and traffic management since the last decade, while artificial intelligence has been applied to the context of traffic at least since 1980 (as stated in `Bazzan and Klügl` [29]). Due to modern life and organization of traffic systems, there is an increasing complexity which can be modeled and simulated with agent systems. With those agent systems, not only the macro view can be modeled, but individual choices even more so. In particular, decentralized use and goals can be researched for a more efficient and emergent system. Applying multi-agent technology to traffic scenarios is extraordinarily challenging, but appealing in order to answer the question of how to model and improve traffic systems at the micro and macro level. Moreover, traffic scenarios are perfect sandboxes for MAS coordination and adaptation models and methods. Agent-based approaches are well established and suit traffic management problems with given geographical, functional and temporal distribution of data and control, but also flexible and frequent interaction between traffic participants and their environment.

Research into agents and traffic is progressing enormously as summarized and discussed in `Chen and Cheng` [62] and `Bazzan and Klügl` [29]. The international workshop 'Agents in Traffic and Transportation' discusses achievements and challenges. In order to maximize the efficiency of the urban traffic system, all traffic components are expected to cooperate.

Besides static and dynamic traffic control, the agent entities of intersections and vehicles are addressed in the following subsections.

3.1.1 Established Traffic Control

Most of the optimization methods target traffic control. Some established approaches of static control of traffic signals are described in the following together with their strengths and drawbacks.

In centralized TMCs, strategies for the improvement of traffic flows are designed area-wide for cities. Thus, centralized control is exerted e.g., using traffic lights, (variable) message signs and other traffic control infrastructure. Examples of classical (pre-designed and off-line) traffic control methods are TRANSYT [293, 369], SCOOT [179], and SCATS [233]. There are multiple limitations to the centralized approach. The most serious disadvantage is the collection and maintenance of all traffic information in one centralized place at the calculated time. This results in large amounts of information to be transmitted to the centralized TMC and the need for a big storage space. This is why it is done in a pre-defined area based on historic information in an off-line modus. Another problem is the necessity of making decisions for the control of big areas on the basis of expert knowledge or extracting big data. Centralized traffic flow optimization requires huge computational power, which may be a limiting factor for TMCs.

Traffic management can be distributed and therefore divided into three levels like in Figure 3.1, with an upper strategic level, a tactical network level (also referred to as MacroBALANCE), and the local optimization of an intersection, which supports the level architecture derived from Sanderson in Figure 5.2. Each level of the BALANCE system architecture in Figure 3.1 has its particular components for making decisions. Whereas in lower levels the processes can be automatized, the strategic level uses an operator interface requiring expert knowledge and intelligent computer support with prepared data. On the local level as seen in Figure 3.1, microscopic data is detected (with FCD data or data collectors) and then processed for short term prediction: according to flow profiles and average values, the data is aggregated and turning movements extracted from the traffic flow. This data is given to signal groups of traffic lights and the aggregated data used and optimized for the local intersection and its time frame. For the general objective, function weights are applied to the data in order to give priorities. On the tactical level, similar processes to the local level are done with the exception that aggregated and turning data is combined and OD matrices are determined. The other steps for traffic signals are done for a local network with optimization and operation weights.

TRANSYT is an optimization of traffic signal controllers in an off-line fashion [293] and generates optimal coordinated plans for fixed-time operation. TRANSYT is used for designing, modeling and optimizing individual isolated junctions as well as large and complex networks. Using SCOOT within TRANSYT, SCOOT values can limit cycle times and, using SCATS, import and export are enhanced. Available as optional add-ons, TRANSYT has the capability to

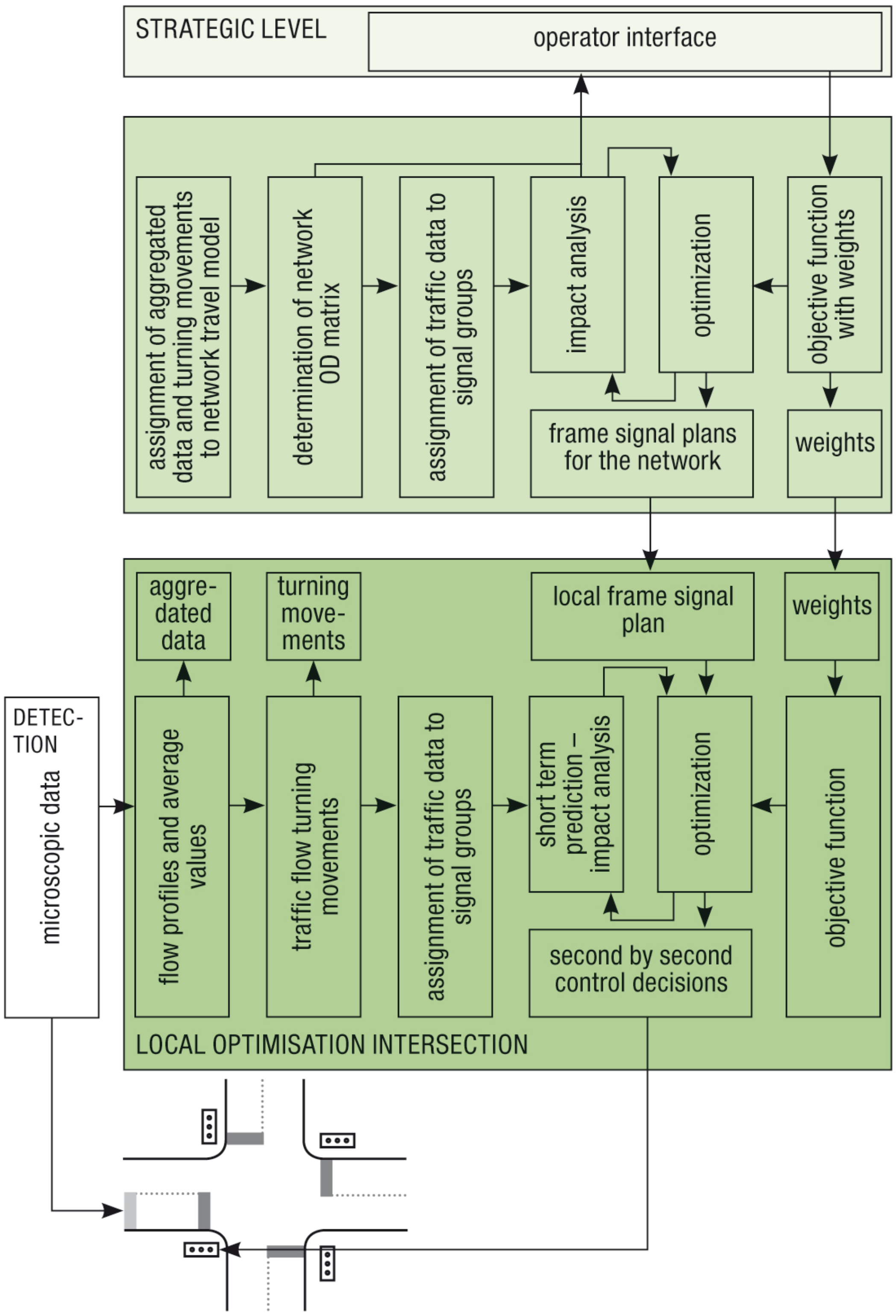

STRATEGIC LEVEL
operator interface
assignment of aggregated data and turning movements to network travel model
determination of network OD matrix
assignment of traffic data to signal groups
impact analysis
optimization
objective function with weights
frame signal plans for the network
weights
aggre-dated data
turning move-ments
local frame signal plan
weights
DETEC-TION
microscopic data
flow profiles and average values
traffic flow turning movements
assignment of traffic data to signal groups
short term prediction – impact analysis
optimization
objective function
second by second control decisions
LOCAL OPTIMISATION INTERSECTION

Figure 3.1: BALANCE System Architecture by `Clark et al.` `(Friedrich)` [66] p. 21/83.

optimize and coordinate traffic signals for microsimulation networks created using AIMSUN and VISSIM. The main drawback is that plans are based on historical data and, thus, computed for a static situation.

SCOOT, Split Cycle and Offset Optimization Technique [179], is similar to TRANSYT, but makes use of buried detectors and is therefore, to some extent, traffic-responsive. A SCOOT system consists of global control on one central computer collecting information of many regions with each including a number of intersections. With SCOOT, the entire network is modeled, and therefore, requires no fixed plans. Similar to TRANSYT, SCOOT uses up-to-date detector data for modeling online the cyclic flow profile. The available data includes flows, delays and occupancy, which is received by one detector per segment, in an upstream location. Thus, SCOOT can monitor congestion in the network. SCOOT models information only once such as the travel time from detector to stop-line. Using the normal SCOOT loops, optimization is based on the real line at the signal and, therefore, is more responsive to unpredictable arrival flows. SCOOT has been installed in many cities. Five cities took part in an assessment of SCOOT. These results show the benefit of an overall reduction in delay of about 12% compared with good fixed time plans. One conclusion is fixed time plans become out of date, then the benefit of SCOOT would increase. In summary, the SCOOT simulated system is more effective than the TRANSYT system. However, the difference between the SCOOT system and TRANSYT as a signal system declined as the traffic volumes approached saturation.

SCATS - Sydney Coordinated Adaptive Traffic System [233] is also based on real-time data like SCOOT, but has a hierarchical structure. Instead of one central computer, SCATS consists of a multiple of regional computers in order to control the traffic sub-systems. Each sub-system is a collection of intersections, of which one is designed to be the critical intersection. The signal time and phase distribution are calculated for this critical node. Based on these calculations, the plans are modeled for the other intersections. Thus, SCATS needs to identify the critical node in each subsystem with plans to use at these intersections. Local adaptation are made upon plan selection. Often deriving these plans is extremely time consuming. In order to maintain satisfactory operation those plans are updated regularly. Typically, subsystems contain 1 to 5 intersections. Vehicles are detected by two types of detector - one near the stop-line to measure flow and occupancy and (sometimes) one upstream, to measure the line of waiting vehicles. Based on the flow in the previous signal times, SCATS calculates green phases. Therefore, unpredictable arrival flows can cause problems for signal phase adaption. The Australian Research board compared TRANSYT and SCATS and found no significant reduction in travel times. There was, however, a large reduction in the number of stops (around 9% in the central area and 25% on the arterial roads).

TRANSYT, SCOOT, and SCATS are centralized commercial systems and have their strengths in their reliability and robustness. These systems are deployed in established traffic control. The weakness is the assumption underlying their simple model that there is a 'typical' level of service (compare `Transportation Research Board` [43]) as a default value for creating their signal-timing. But, in reality, the LOS has many variations including the peak rush hours or the nighttime shortage of traffic. Additionally, there is the issue of costs and the maintenance of traffic management centers (TMC) for collecting, computing, and storing the data.

3.1.2 Dynamic Traffic Control

Dynamic traffic management targets and optimizes the control of traffic flows in the existing networks with infrastructure elements like traffic lights and detectors. Most approaches to dynamic traffic management follow a centralized perspective, collecting data through sensors, detectors and FCD data and processing.

Similar classic software, but decentralized frameworks, are mentioned by `Bazzan and Klügl` [29]: PRODYN [166], OPAC [120, 121] and UTOPIA [246].

PRODYN is a real-time algorithm computing the best signal phases integrating the delay for any traffic network. It is a hierarchical algorithm using Forward Dynamic Programming (FDP) to compute controls at the lower level (intersections) and decomposition coordination techniques at the upper level. PRODYN attempts an explicit minimization of total delay. The implementation relies on the use of a network structure of microprocessors, the tasks of which are optimization and state estimation.

OPAC stands for Optimization Policies for Adaptive Control and is a computational strategy for traffic signal control responding to real-time demand. OPAC features performance results which are close to the theoretical optimum while requiring online-data from detectors. OPAC uses the dynamic programming approach as a mathematical technique for the optimization of multi-stage decision processes. OPAC minimizes vehicle delays and the percentage of stopped vehicles. Existing microprocessors can be used for implementation and it forms a building block for decentralized control in a network.

UTOPIA is an outline of applied control strategies and hierarchical, decentralized traffic light control. The main features of the system are priority assignment to selected public vehicles and private traffic optimization. To do this, UTOPIA uses closed-loop control strategies starting from a global approach and then decomposing into hierarchical decentralized units and applying the rules for the intersection with appropriate techniques and algorithms with a strong interaction concept. On one hand, it takes neighboring intersection states into account in order to build a dynamic signal coordination and, on the other hand, it is constrained by limits given by the area level control in order to remain sensitive to traffic-dependent criteria.

TUC stands for Traffic-responsive Urban Traffic Control [85] and uses a variation of the store-and-forward control method to model the traffic flow and compute control policies efficiently. In 1997, TUC was initially developed and field-implemented in Glasgow, Scotland, within the European DRIVE III project TABASCO (Telematics Applications in BAvaria, SCotland and Others) as part of the integrated traffic responsive urban corridor network control strategy IN-TUC (INtegrated - Traffic responsive Urban Control). The objective of TUC is the homogeneous utilization of the capacity in urban networks. In urban signalized intersections the green phases are manipulated appropriately while the cycle length stays unchanged for reaching the mentioned control objective. In TUC all the control decisions are based on real-time data measurement which are collected from detectors. TUC is built upon well-known methods of automatic control and optimization theories. This allowed the development of a strategy that is robust with respect to measurement inaccuracies so as to be able, even in cases of insufficient data, to react correctly to the current traffic conditions, and simple, so as to permit the execution of all required calculations in real time. In spite of its centralized functional architecture, TUC lacks flexibility regarding modifications and expansions of the controlled network.

At the University of Arizona RHODES [251] a **R**eal-time **H**ierarchical **O**ptimizing **D**istributed **E**ffective **S**ystem was developed by a research team. The 'dynamic network loading' model is the highest level capturing the slowly varying characteristics: the network geometry and the typical route selection. RHODES distributes the duration of green to each demand pattern and phase based on the traffic load on each intersection at the middle level of the hierarchy, referred to as 'Network flow control'. Concluding from the approximate green times an appropriate phase change is selected based on arrivals of individual vehicles. This is the third level, the 'intersection control' at each intersection.

CRONOS [44] is a real-time urban traffic control algorithm. The centralized control strategy encompasses a heuristic global optimization method with polynomial complexity. Although the price of specifying a local minimum, this allows simultaneous considerations of several intersections. Benefits are obtained on the total number of stops, especially in comparison with a local strategy. The general behavior results in more global green duration and a higher average number of cycles per hour..

TRYS [73] is a real-time traffic management application which is based on a knowledge-based environment. From the knowledge level to the symbolic level, the building process is a progressive definition of knowledge features. The problem is shortcomings on traffic management systems. A description of the tool is commented on supporting the organization of structured models at the knowledge level. The generic model, intended to deal with traffic management, is described and modeled for Barcelona.

SCOOT, SCATS, PRODYN and OPAC are characterized by online signal timing with adaptive features and all require one detector for each link, whereas PRODYN uses two detectors per link. But the principles upon which the four algorithms depend vary from cyclical to acyclical, and centralized to decentralized. SCOOT and SCATS seek to adjust cycle time, phase split, and offset so that stops, delays, and line up lengths are minimized. By contrast, PRODYN and OPAC attempt to find optimal acyclical settings for traffic- responsive, coordinated control. PRODYN and OPAC follow the decentralized paradigm and perform optimization at each intersection through dynamic programming techniques and a rolling horizon.

More advanced model-based traffic-responsive strategies like OPAC, RHODES, PRODYN, CRONOS, and UTOPIA formulate the traffic-responsive urban control problem as a combinatorial optimization problem, and, with the exception of CRONOS, they employ exponential-complexity algorithms to solve for a global minimum. But these strategies do not consider splits, offsets, and cycles explicitly.

An alternative to the classical form of control discussed so far is decentralized traffic management, in which traffic participants such as traffic lights act according to local information, i.e., they optimize their behavior based on the available local data. This avoids the transmission or storage of big data. Traffic control elements perform a local control of neighboring streets and a limited number of intersections, for example in `Pohlmann` [274]. A new Adaptive Traffic Control Systems (ATCS) prototype has been developed and evaluated by `Pohlmann`. ATCS control a set of connected intersections and aim to optimize traffic signal control in real-time. Depending on the current traffic state, the signal plans adapt.

However, the main disadvantage of this approach is that the regulation and the optimization are done only locally and on the basis of available data; this means that no (global) guarantees of optimality can be provided.

Priemer [276] makes use of the technical advances in wireless communication and GPS-location. Thus, vehicle detection at signalized intersections can be done in different ways. Within the communication range, approaching vehicles can be detected early and continuously. The decentralized control method of traffic signals is based on vehicle-to-infrastructure communication, which enables the traffic lights to adjust the green time and phase sequence immediately to the current traffic situation.

Tomforde [346] provides an adaptive approach with methods of Organic Computing (OC) to combine upcoming visions with existing technical systems. Self-configuration and self- improvement is realized in an architectural framework and tested within mobile communication with and increased 6% for available bandwidth and urban traffic network on the parameter of delays, which can be improved up to 16%.

The interplay of advanced sensor systems, large-scale intelligent data processing, and ubiquitous mobile networks are presented in works by `Fiosina` [104, 105], additionally transports the vision of the Internet of Things to the traffic management domain: in the digital city, digital models of traffic infrastructure and traffic participants will be maintained in real-time, and traffic management systems will make decisions based on these models.

Self-responsibility and the gathering of various pieces of information change the former centralized structures. New reward systems with a mix of individual, group and global results need to be investigated. Innovation trends like vehicle-to-vehicle (V2V) communication affect traffic management systems, creating new challenges and opportunities. Vehicle-to-infrastructure (V2I) and V2V communication,collectively referred to as V2X, enable real-time data exchange and coordination among traffic infrastructure and vehicles.

From the decentralized perspective, with the development of advanced assistance functions such as dynamic navigation and adaptive cruise control and intersection assistants as mentioned by `Baskar et al.` [23], as well as autonomous driving support described by Kang et al. [199], vehicles themselves become more ’intelligent’ and more autonomous.

3.2 Agent Modeling and Simulation in Traffic

Agent modeling and simulation in traffic is the second category of autonomous vehicle group formation discussed here. It enables various zoom levels onto points of interest such as microscopic behavioral details, mesoscopic areas of interest like a city district or the macroscopic view, such as how the overall performance of the traffic system is. Dealing with traffic simulation bears many challenges concerning the necessary knowledge and level of detail, spatial distribution, the integration of different goals present in traffic systems and the phenomena of emergent behavior. While simulations for central traffic management precisely depict all potential changes of parameters of the overall state of the traffic, organic computing and Multi-agent Systems try to combine common jointly occurring changes of state parameters, so-called patterns, into explicit modeling constructs. In this way, the depiction of larger simulations becomes clearer.

The agent metaphor reproduces intelligent decision-making entities and therefore offers many advantages compared with classical approaches in traffic simulation which can also be modeled through MAS. In Multi-agent Systems, dealing with heterogeneous and variable structures is possible, and complex information processing and decision making with dynamic information and multiple factors can be enabled. Thus, also group behavior, learning and adaptation can be modeled. This paradigm supports distinct and sophisticated visualization and permits modeling heterogeneity on different levels. The specific focus is on the enactment of autonomous processes in traffic in the depicted simulations through MAS. Similar to formal models, group formations, thus, need well-defined execution semantics. Moreover, they need to include information such as which agents are authorized to perform certain tasks of the process or how to handle the input and output of the various activities.

According to Schmeck et al. [309] organic computing assumes collections of autonomous systems in our daily more and more computer-aided environments. In those (future) systems, sensors and actuators observe and act their environment including free communication. Additionally those organic systems organize themselves for performing the actions and services that seem to be required. Organic computing makes use of intelligent software systems using technology for observing the environments and taking actions within.

Thus in our environments, the networks of intelligent systems become more present. On one hand, they open new application areas, but, on the other hand, bear the problem of their controllability. Hence, the construction of such systems should be as robust, safe, flexible, and trustworthy as possible. In particular, a strong orientation towards human needs seems absolutely central as opposed to pure implementation of the possible technology. In order to achieve these goals, the technical systems need to create lifelike properties for acting more independent, flexible, and autonomous. Those systems are called „organic“. Hence, an „Organic Computing System“is a technical system which adapts dynamically to exogenous and endogenous change. Thus, organic computing usually abstracts from implementation details.

Multi-agent Systems (MAS) enrich the environment, including traffic control, with additional, precise information, which is required by decentralized systems to execute the process.

In this context, theories and models of autonomic [201] and organic [383] computing come into play, reflecting the abilities of a system to organize or manage itself according to high-level goals without explicitly being commanded to do so. In the computer science literature, these properties have been condensed to the term self-X, incorporating self-configuring, self-healing, self-optimizing, and self-protecting. A unique class of systems to embrace and support the design of autonomous systems are multi-agent systems (MAS), i.e., systems that are perceived as consisting of a set of autonomous intelligent components defined as agents, which are capable of flexible and coordinated action in order to achieve their design objectives. Autonomous MAS [340] extend the notion of autonomy to loosely coupled, decentralized systems, and are a promising modeling approach for autonomous traffic management systems (TMS).

Traffic control provides opportunities for cooperating agents. The paradigm of agent technologies enables, besides modularity, also modeling and simulation of management strategies. Traffic control usually coordinates traffic lights but can also focus on **vehicle cooperation**. In the following, the focus is more on integrated environments for traffic simulation and control in order to improve the management of the traffic system as a whole. Challenges are: allocation of limited resources, and the choice of organizational models for hierarchical, classical authority-based approaches versus totally decentralized ones, which show less commitment to global performance.

3.2.1 The Macroscopic View

Within classical AI, distributed AI (DAI) addresses group behavior with highly cognitive agents, but those are not embodied nor situated in a simulated or natural physical world -an overview is provided by `O'Hare and Jennings` [265]. Another branch of DAI deals with simple distributed systems with the focus on cooperation and competition in multi-agent environments, see `Jennings` [187] and others.

Jennings [187] approaches group behavior on a high level:

> Joint responsibility is a new meta-level description of how cooperating agents should behave when engaged in collaborative problem solving. It is independent of any specific planning or consensus forming mechanism, but can be mapped down to such a level. An application of the framework to the real world problem of electricity transportation management is given and its implementation is discussed. A comparative analysis of responsibility and two other

> group organisational structures, selfish problem solvers and communities in which collaborative behaviour emerges from interactions, is undertaken. The aim being to evaluate their relative performance characteristics in dynamic and unpredictable environments in which decisions are taken using partial, imprecise views of the system.

An urban traffic management system based on agent-based distributed and adaptive platforms are both feasible and effective with mobile agent technology. However, enormous computing and power resources are required for the large-scale use of mobile agents. Additionally a complex, powerful organization layer is needed. To deal with this problem, a prototype urban traffic management system using intelligent traffic clouds is proposed for future work.

3.2.2 The Mesoscopic View

An alternative mesoscopic tool is the game theory metaphor, in which the interaction is formulated as an abstract game where all individual decisions influence the overall outcome. The analysis is based on payoff matrices or functions which describe possible outcomes for different participants. Therefore, equilibrium states can be identified and a prediction of rational actors can be selected. Agents choose one alternative of game-theory analysis like minority or congestion game and the less-populated option wins or receives the highest reward. After some game repetitions, agents will try to avoid selecting the majority alternative. Therefore, those games provide a good abstract model for many traffic scenarios and are also modeled with the agent paradigm. However, agent-based simulations have more modeling variables such as irrational or malicious behavior, decisions based on a lot of information and dependencies on resources or other agents.

Queueing simulation as in `Charypar et al.` [61] is an intermediate solution between microscopic and macroscopic simulation paradigms because it abstracts the actual decision making while driving by capturing the time that a driver stays on a particular link based on its position in a queue. Vehicles move passively in the system because no active decision making is done during driving. This approach is very efficient and is integrated in the agent-based traffic simulator MATSim[1]. With the lining up approach, actual movement, with agent-based demand generation and departure time, mode and route choice, is simulated.

In `Wang` [370], a real-life transportation system is supported by an artificial equivalent in TransWorld with a feedback control mechanism. Modes are the key for connecting the two worlds. In the learning and training mode, the artificial system serves as a data center for learning operational procedures and is loosely coupled to the real world. For experimenting and evaluation, there is a mode where the artificial world is the platform for conducting experiments and for predictions regarding the actual system. But, for control, both systems need to be tightly connected and the artificial system replicates the actual behavior. The

[1] `www.matsim.org`

AI-based system replaces the analytic reference model and is a generalization of adaptive control. Coupling both systems in parallel seems to be a bottle neck.

Another artificial transportation system is modeled by `Rossetti et al.` [298]. It is based on agent programming and can therefore deal with social and behavioral models.

Regarding autonomous vehicles, a management system is proposed by `Bazzan et al.`, which includes [27] an agent and market-based approach where a fleet of automatized guided personal rapid transit vehicles are managed. Some variants are discussed: centralized versus decentralized, with or without en-route re-planning, and action-based versus first come, first served.. In some cases, the centralized service is feasible, but communication, reliability and fault-tolerance are important issues for investigation.

The work of `Vasirani and Ossowski` [363, 364] deals with networks of intersections with a market-based mechanism where vehicles trade with infrastructure agents in a virtual marketplace. They cross intersections with reservations they purchase. The goal is to combine the global benefit with the social welfare, e.g., average travel time. To this end, marketplace rules are designed. The scientific findings are that, with the same traffic throughput, the individual vehicles improve their average travel time, when the costs of the infrastructure are low. M.I.T.E. [362] - Multiagent Intelligent Transportation Environment was developed for this research.

3.2.3 The Microscopic View

Multi-Agent Systems (MAS) [268] [382] are a promising approach to model and simulate traffic systems and offer many advantages compared to conventional traffic simulation. Heterogeneous and changing structures are handled elegantly by MAS, complex information processing and decision making can be done while considering multiple factors and dynamic information (this enables grouping, learning and adaptive behavior), and it supports distinct and elaborated visualization while also permitting distinction between the micro, meso and macro perspectives in simulation. Thus, heterogeneity can be modeled on different levels based on contexts, parameter sets and architectures.

Th above-mentioned agent features distinguish agent-based approaches to traffic simulation from more conventional microscopic ones.

In `Ortúzar and Willumsen` [266], a model with four steps is proposed with a phase for initialization, followed by an assignment phase, a mobility simulation and decision making. The initial phase collects data and manipulates the number of trips between the locations. A trip is connected to a route in the assignment phase and results in the simulated load on the network. For determining the actual traffic flow data and, depending on the focus of the study, a mobility simulation is done. Decision making is involved across all aspects and can model explicit individual decisions with reasoning about agents' behavior in terms of actions, locations and times. Therefore, the individual decisions are context-dependent.

With weights and parameters for modeling agent decision making through an individual function, the work of `de Palma et al.` [77] is based on econometrics. Alternatives are evaluated and optimized in order to minimize the gap between stated measure and simulated distribution. The contribution of this work is that, through the weights, the preferences of the traffic participants are modeled. The authors' criteria are: simple functions for estimation, input data of a certain quality and results which should be able to transfer to similar scenarios.

Concepts of agents in traffic are formulated in `van Katwijk and van Koningsbruggen` [359], especially with heterogeneous agents and system components. BDI agents were suggested from a modeling perspective to represent and reason about the agents' knowledge. A coordination protocol is presented for cooperative agents that discusses a priority in resource allocation . This helps the distribution of vehicles in order to manage traffic flows. Resources of the vehicles are the allowed speed and the vehicle flows. Resource-bound decision agents manage and keep track of the amount, times and priority of requests and their changes in order of priority or non-granted requests. Problems the authors discuss are conflict resolution and local versus global performance;these remain partially unsolved. The latter can be solved by this thesis with a hierarchical organization where authority does both: solves conflicts and pursues global performance. Other hierarchical approaches also tackle this problem, like [76, 116, 297].

A testbed for traffic management which deals with different complexities, diverse policy goals and various forms of traffic problems is presented in `van Katwijk` [360]. The authors present three components: interaction, intelligence and the world for the implementation of the decentralized traffic control. The interaction model uses mainly communication for the interaction among the agents. The intelligence models are rule-based and Bayesian interference is used for the uncertainty agents can have. Although the work was aimed at decentralized traffic concepts, it seems that it suits the global perspective more for hierarchically organized controllers and for inter-controller coordination. In order to change this approach, different kinds of organization controllers are needed, such as intra-controller coordination among the phases of an intersection and other forms of coordination not just based an pure communication.

In the PhD thesis of `Gershenson` [128], a multi-agent simulation is used to study three novel self-organizing methods without direct communication, which are able to outperform traditional rigid and adaptive methods. The focus is on traffic light synchronization and adaptation to changing traffic conditions. Improvements in reducing waiting times and the number of stopped vehicles, and an increase of average speed are shown in a realistic simulation of a Brussels avenue. Self-organizing methods outperform the current green wave method. In the modeling, the relevance of vehicle groups is described in `Gershenson` [128] (p. 65) such that

> as more cars join the group, cars will be made to wait shorter periods before a red light

and vehicles moving jointly together improves the traffic flow. The approach of `Gershenson` achieves maximal satisfaction for the city, the traffic lights and the vehicles, but states that this situation is unrealistic. Due to his toroidal topology of the simulation environment, full synchronization is achieved. His method gives the best performance with low traffic densities; with high traffic densities the method proves to be highly inefficient because traffic lights constantly switch their signals. In his discussion it is pointed out that, in his methods, vehicles are not forced into groups, but induced, which gives adaptation possibilities in order to have a flexible system.

`Baber et al.` [12] relate to the mechatronic level and give insights in cooperative driving robots and driving maneuvers with lane following, cooperative overtaking, unsignalized intersections, distance and tracking control, and dynamic obstacle avoidance algorithms. In contrast to this thesis, it includes the sensory aspects as well as cooperative algorithms, which are mostly included in the traffic simulators. Cooperative overtaking is employed in the section describing ATSim in Chapter 4.

Agents are situated in the environment and pro-actively pursue their goals. Thus, `Barber and Park` ([19] p. 6) describe autonomy as relational and limited. An agent's autonomy is meaningful in relation to the influence on other agents but also limited by the relationships. Autonomy is important to an agent's belief in associated goals, such as the degree of dependence on others or the degree of controlling its beliefs. Beliefs are constructed from information about the environment and the situation, also acquired from other agents. The agent's autonomy of beliefs is key for accurate belief formulation and goal achievement.

According to `Bradshaw et al.` ([19] p. 20), there is a descriptive and a prescriptive dimension of autonomy. The first describes horizontal actions which an agent in its context is capable of performing and the second dimension represents the vertical actions which it is allowed to perform or which it must perform due topology constraints. Therefore, autonomy comes in pairs of actions which can be categorized into potential, possible and performable and their counterparts permitted, available and achievable. WHereby, different settings of 'absolute freedom' (= potential + permitted), 'absolute capability' (= possible + performable) and 'absolute autonomy' (= absolute freedom and capability) are described. There are many other dimensions of autonomy [1], [235] [241] including `Castlefranchi's` dimensions [60] of the agent's autonomy, which derive from its architecture and from the theory of action.

`Russell and Norwig` ([300] p. 704) describe learning as follows:

> An agent is learning if it improves its performance on future tasks after making observations about the world. Learning can range from the trivial, as exhibited by jotting down a phone number, to the profound, as exhibited by Albert Einstein, who inferred a new theory of the universe.

For real-life applications learning [7] is essential for designing and constructing autonomous software systems. Especially in dynamic and complex domains, autonomy and adaptability (a form of learning) are closely related, because agents make their own decisions if they are provided with the ability to adapt to the changes occurring in the environment they are situated in.

In the interdisciplinary debate `Kühn et al.` ([5] p. 1) discuss that

> both learning and adaptation are connected with something that changes, may it be a person, an abstract agent, a system or any kind of object.

Adaptivity is changing the internal structure of individuals or collectives and is - in contrast to learning - of a passive nature. Learning must be in the form of a manageable object which controls the behavior: a pattern, circuit, expression, rule, program or the like. `Kühn et al.` also state that intelligence and learning are related, and attempt to understand this two-fold connection by the investigation of learning and then adding the outcome to the contribution of intelligence. They also observed that learning is associated with improvement, that is, after a learning process the actions are performed in a better way by the learner; this was especially true for supervised learning.

Reinforcement learning has been defined as mapping situations into actions in order to maximize the numerical reward. In contrast to supervised learning such as pattern recognition or artificial neural networks, where the learning agent is told what to do, in reinforcement learning agents must discover which actions yield the most reward by trying them. Typically, actions may affect the immediate reward, but also the next situation, and, through that, all subsequent rewards. The most important features of reinforcement learning are trial-and-error search and delayed reward. Temporal-difference and Q-learning are solving the problem of finding an optimal policy, i.e., the optimal actions at a given time defined as mapping from perceived states of the environment. For further reading see [198] [334].

3.3 Interaction: Traffic Communication

This represents the third category and focuses on direct and indirect communication with approaches supporting interaction and the exchange of information. Vehicle-to-infrastructure (V2I), vehicle-to-vehicle and vehicle-to-eXchange (V2X) communication are new technologies actively entering the traffic domain. Vehicles need to be equipped with communication technologies and systems in order to be able to exchange and process the information exchanged. Communication technologies are not the focus of this thesis. Nevertheless, communication is needed to form groups. Therefore, agent-based communication is discussed in the following with the focus on intersections 3.3.1 and vehicles 3.3.2.

Nowadays, the demands of increasing (urban) traffic can be met by the classical idea of creating additional capacity by building roads; but new emphasis is being placed on the improvement of existing infrastructure in order to utilize the available capacity in better economical, environmental, or social ways. Interaction and communication can help distribute the traffic flows in the network through information exchange.

3.3.1 Intersection Modeling

Intersection agents regulate the node of joining streets. Most common are traffic lights, but `Dresner and Stone` [89] propose a reservation-based system specifically for intersections in order to alleviate traffic congestion, which causes lost productivity and a decreased urban standard of living. They assume that the cars are controlled by agents. First, a custom simulator is described. It was created to measure the different delays connected to the moving traffic through an intersection. Second, a quality metric evaluates the traffic control at an intersection. The reservation-based system, using the simulator and the metric, performs, according to the authors, two to three hundred times better than traffic lights. The intersections simulator and MAS control policy described reach the theoretical optimum in simulation. Current limitations are margins of error in the system that are too small for a human to be able to control the vehicle. Due to the agent concept, no mixed traffic or human drivers with different penetration rates are executed. There is the the idea of only a few human drivers mixed in with the autonomous agents; in this case a signal could be forwarded to the intersection that a human driver is approaching. For large amounts of human drivers, a traffic light is more appropriate. Once autonomous vehicles are common, real traffic could be managed with this mechanism. The autonomous vehicles are limited in turning and changing velocity while in the intersection. `Dresner and Stone` [92], [317] extended their AIM approach to apply to autonomous vehicles as well as human drivers sharing the road. This is still under active investigation and more details on the decentralized architecture are addressed in the organization section 3.4.2. Those works in the AIM context are a good source because they share the assumption that vehicles are controlled by agents and they inspire metrics for evaluation. However, this thesis does not focus on reservations or market-based approaches, but uses the urban traffic as-is with traffic light regulations, and limits the street use to autonomous vehicles only. In future this thesis should be extended to sharing urban traffic with human drivers.

`Kolodko and Vlacic` [209] describe a intersection control system which is similar to the reservation system with granularity of `Dresner and Stone`. Successful implementation on real autonomous vehicles has been reported.

Another work without traffic lights was done by `Mandiau et al.` [240] a single intersection with only stop signs, thus, **lightless**, is proposed as a two-player game of autonomous vehicles arriving or entering the intersection. The actions of those vehicles are to move or to stop and priority rules are created for the type of game and the matching payoff. Tests show the actual flow of vehicles of two and three agents in complex, highly conflicting scenarios. An open question remains as to how to implement those mechanisms for a network of intersections. The game-theory approach of autonomous vehicles at intersections is interesting, but not the focus of this thesis, although the scenario of an intersection is also used as a test case.

Intersection management has, in most cases, been studied from a theoretical perspective, whereas the AIM project, presented above, is a counter-example.

Theories reduce the problem to scheduling jobs that can compete for mutually exclusive resources. Irani and Leung [182] create ’conflict graphs’ in which the nodes, which share an edge, compete for the same resources and search these graphs for independent sets. A probabilistic approach is used and actual simulated results are reported. Their approach aims to tune the underlying system of traffic signals more effectively. The idea of approaching the subject via graph theory is also used for this thesis.

Most prior work with multi-agent approaches has focused on decreasing delays and increasing the throughput for traditional traffic light systems, just as this thesis does as well, with the focus on the vehicles instead of on the traffic lights. One significant example is presented by Roozemond, which allows intersections to act autonomously, sharing the data they gather [296]. The intersections use this information to make short- and long-term predictions about the traffic and adjust accordingly. However, this approach still assumes human-controlled vehicles and the agents in this case are only the intersection controllers themselves. It could be a good idea for the continuation of this thesis to group also intersections in a market-based approach in order to give guarantees to vehicle groups.

Hierarchical organization of agents is a coordination technique that uses some sort of organizational structure where a hierarchy of authority exists. Instead of communication, organizational structure can coordinate multiple agents with less coordination than the above-mentioned pure communication approaches. Roozemond [297] investigates in his research whether autonomous agents are deployable for substituting traffic management centers (TMC) and shows how they are useful in urban traffic contexts. He proposes a model with intelligent traffic signaling agents (ITSA) for intersection control and for road segments and also agents which have authority. The coordination is based on authority agents, which supervise and control several intersection agents who are in charge of managing the control of the intersections. Intersection agents use predetermined rules for selecting the control strategy and a prediction model for future estimations and they are able to communicate with neighbors. The intersection agent calculates signal plans, checks with neighboring agents and, based on predictions and rule sets, plans a signal control strategy. Non-conflicting cases can be handled by the intersection agents whereas, in case of conflicts, authority agents need to intervene. Agents select their choice on the base of the highest local efficiency and, *in favor of cooperative behavior, intersection agents need to negotiate and can sacrifice some performance*. Here, an advice mechanism is proposed. The proactive behavior of agents provides better usage of the intersection capacities and traffic conditions can be handled in real-time in a stable and integrated fashion for the overall traffic system.

Deriving from the area of mechatronic systems, agent-based controllers were designed to be

> complex technical systems whose motion behavior is actively controlled with the help of computer technology,

as described in Oberschelp et al. [264] (p. 1), pointing to one application of such controllers: the intersection management.

Another interesting self-organized control approach is done by `Helbing et al.` [163]. Here, traffic patterns are observed and synchronization implies the emergence of green waves. Adaptive strategies of traffic light coordination and local interactions between vehicles and signal lights are accomplished through self-organizing principles. Vehicles can synchronize traffic lights and organize green waves. One important conclusion that is drawn is that *forming vehicle platoons or clusters in front of changing traffic lights can increase the capacity of an intersection and is most efficient if arrival rates are reduced*. But instead of forming individual traffic groups, the authors choose to create vehicle platoons by means of traffic lights and argue that benefits come especially from the sum of arrival flows exceeding the capacity of non-signalized intersections with incompatible flows. One *suggestion is to use distributed, local control instead of traffic management centers (TMC) in favor of greater flexibility and robustness, and that travel time information can be transmitted by mobile communication to enhance route choice decisions*. To solve the problem of lining up networks such as online production scheduling.

3.3.2 Vehicle Modeling

Vehicle agents are investigated for simulation models, or as an in-vehicle software agent. Real world autonomous vehicles are still being developed, such as Google Car [2] or iCar [3].

According to `Jiang et al.` [193], vehicle agents can have the objectives of shortest route and congestion avoidance. Therefore, the vehicle agents leave intersections with a high density in order to avoid congestion. The internal degree of freedom is reflected by the choice of the next action and its dynamics. Clustering vehicle agents into groups would lead to an increased density which causes others to avoid the intersection and thus alleviates network congestion.

Hybrid systems with an in-vehicle software agent and a driver can improve a driver's psychological state and increase road safety according to `Jeon et al.` [192]. The study explored the possibility of using an in-vehicle software agent using 60 undergraduates in a simulator with two types of agent intervention. On one side driver situation awareness and driving performance are improved by speech-based agents, and on the other anger level and perceived workload is reduced. The practical implications include guidelines for the design of social interaction with in-vehicle software agents.

3.4 Organization: Traffic Architecture

The organization in form of traffic architecture, as the forth category of autonomous vehicle group formation, provides practitioners with the opportunity

[2]`https://waymo.com/`

[3]`http://www.macworld.co.uk/news/apple/icar-apple-car-release-date-rumours-news-caros-evidence-concept-images-patents-2017-3425394/`

to adopt the traffic control in different dimensions as a basis for discussions. Those dimensions are top-down as a central approach and bottom-up as the decentralized one. As a third middle dimension there are hybrid architectures, which have decentralized entities controlled by a hierarchical upper entity. Since the classical view of traffic systems is central, one line of research is to perform a headstand. Therefore the perspective of my research focuses on decentralized traffic coordination and cooperation. Interesting research domains which can deal with complex traffic control on the base especially of decentralized architecture are the fields of Organic Computing and Multi-agent Systems. Thus, multi-agent architectures for traffic control are analyzed here with the focus on being suitable for autonomous vehicle group formation.

Coordination promises improvements of performance in networks of intersections and is an important factor in MAS. Starting from the optimization of single intersections, there are new challenges to meet because agents can ideally coordinate with others, but can also ignore other agents. This could result in sub-optimal solutions for the macroscopic perspective. Techniques of coordination are described in the following:

- **communication** for exchanging data about traffic states and control actions. Here there is a lack of a protocol which integrates non-local information with local information. For example, what to do with a conflict that arises due to a request from another vehicle agent which is not compatible with the local actions? In this case the agent will try to solve its own local optimization problem.
- **hierarchical organization** of systems or agents is useful for solving conflicts at an upper level. This still needs communication but less than the first approach.
- **self-organizing** systems or agents
- **learning** systems or agents which control their local environments and coordinate their actions
- **lightless** intersections
- **market-based** approaches
- **collaborative driving and vehicle cooperation**

The two coordination techniques of learning or self-organizing agents may be incompatible with the dynamics of the environment, as one potential drawback. As pointed out by `Bazzan and Klügl` [29], especially learning-based approaches can show a poor performance on the global system level unless agents' utility functions are linked to the global objectives of the system. This can be formalized into a general statement which is also valid for non-learning systems: bad design leads to poor performance.

3.4.1 Central Architecture

For global fairness `Balan and Luke` [14] propose history-based controllers. The traffic lights collect information about the vehicles and their recent histories.

Therefore, they communicate with other signaled intersections through the histories of vehicles which travel from one to another. The idea is that controllers give vehicles credits that they receive at red traffic lights and are allowed to spend for green lights when passing through the next intersections. The traffic lights decide on the base of the credits and provide a kind of global fairness for reducing the average waiting time for vehicles on the trip. At the destination, the cars send their average waiting time to evaluate the efficiency of control, but no incentive triggers this behavior. Although these history-based controllers are robust, the authors also comment that this provides more efficiency than fairness.

The approach of `Au et al.` [9] proposes a management policy, which processes several reservations at once after it collects some request messages. This method tries to balance the traffic throughput for all streets and therefore avoids the starvation of secondary roads.

Focusing on the test case of a single intersection, the following works of `Kosonen` and `Koźlak et al.` provide two different approaches.

The work of `Kosonen` [210] studied a traffic signal control system combining multi-agent control scheme, real-time simulation, and fuzzy inference. For that, various agents interact in a single intersection. Indicators from the simulation model are derived as input to the control scheme, which can be used off-line or on-line in simulation. The simulation is also done in a microscopic traffic simulator called HUTSIM developed by the Helsinki University of Technology. Depending on the volume of traffic and the permission of other agents, each signal operates individually as an agent. Hence, the agents negotiate with neighboring intersections about the control strategy. The agents exchange local traffic and control data and the decision making is based on fuzzy inference. This supports the fluency in the network and has other benefits for economy, environment and safety. The paper demonstrates that the challenge is to integrate fuzzy control with coordination of agents in order to solve both the local and the coordinated optimization problems. This combination of integrating local into global coordination is relevant, but not the focus of this thesis.

`Koźlak et al.` [212] presents a multi-agent system for modeling and optimizing city traffic. They proposed a multi-agent system to control traffic signals with intersection managers to predict future traffic conditions. Traffic jams are analyzed, that is, when there is a decrease in average vehicle velocity in the whole city or a part. The two regulating screws are the city light management or the modification of intersection and road network topologies. This thesis also aims to optimize the urban traffic setting, but through forming autonomous vehicle groups based on their position and destination.

`Bazzan` [25] uses methods of evolutionary game theory and stochastic games without explicit communication between the agents. Traffic lights are modeled as agents and have only local knowledge. Those agents execute experiments and receive rewards depending on other neighborhood experiments performed. In the network, stochastic events may happen; those are modeled by mutations. Agents are able to perform better towards the global goal depending on how often the stochastic events take place. The fitness for each strategy is calculated and influences future generations of strategies that are used by the agents. Results show that, with relatively equal vehicles in both directions, the agent-based mechanism performs better due to its adaptability compared to the central progression. Payoff matrices need to be formalized explicitly by the designer of the system and it is time consuming to implement all the different options of coordination. Therefore, a challenge is to develop mechanisms that fulfill the needs of large-scale networks.

While the single intersection scenario is comparatively straightforward, big challenges are associated with involving a network of intersections because there is no obvious way to coordinate the selection of control actions. Coordination can be done individually, implicitly, or with communication. Especially with communication, conflicts can arise and it is difficult to handle them. Hierarchical organization can solve conflicts, but has the effect of reducing the autonomy and can compromise local performance, because decisions are made at the superior level and lower agents need only to execute them.

The domain of traffic light synchronization or green waves can be seen as a problem of the assignment of a coordinated signal plan to each two crossings in a traffic network. Classical approaches like SCOOT [179] and SCATS [233] focus on progression or synchronization of traffic lights on a main artery road mostly in an off-line fashion like TRANSYT [293]. Conflicts may appear to extend the synchronization to a network because different directions compete for throughput. The conventional approach asks traffic experts to solve those conflicts. Alternatives are to shorten the green waves in segments of the network and replace the traditional arterial green wave.

One **communication** approach is to negotiate about which traffic direction gets preference for the throughput. An approach based on distributed constraint optimization problems (DCOP) was proposed in `de Oliveira et al.` [74]. It provides a cooperative mediation solution between flexible coordination with implicit communication and the classical off-line and centralized solution. Each node of the graph has a traffic light. A constraint is that the signal cannot synchronize with neighbors located in different directions at the same time and a conflict arises when neighbors coordinate in different traffic directions. In `Junges and Bazzan` [197], the same scenario is extended for larger networks in order to investigate performance issues such as when an agreement is reached and how many messages have been exchanged. Three algorithms for DCOP are applied to a complex and dynamic context, specifically the synchronization of traffic lights, in terms of complexity and quality of the solution.

Another hierarchical multi-agent system with three levels is proposed by `France and Ghorbani` [116] for optimizing urban traffic. Here, local traffic agents (LTAs) represent intersections and are in charge of the traffic light patterns, while coordinator traffic agents supervise a few LTAs concerning global objectives and information traffic agents handle congestion. A hierarchical model that also uses a global traffic agent is proposed in order to modify the local optimum to improve the global system performance because the authors claim that *coordination is the key to maintaining balance between global and local perspectives*. They point out that the equilibrium between those systems is important. Classical algorithms only focus on the global optimum and other multi-agent approaches only on local optima and, therefore, `France and Ghorbani` propose a power hierarchy in which higher agents observe and can alter the local decisions, leading to improvements in the overall performance.

For improving the robust and well-performing TUC research [85], a model predictive control approach is proposed by `de Oliveira and Camponogara` [76]. It uses multi-agent control for linear, dynamic systems. The model by `de Oliveira and Camponogara` decomposes the predictive control problem of the centralized model into small interconnecting sub-problems with local input constraints that are iteratively solved by the distributed agents located at intersections. The agents sense the state variables and set the values that are under their control. In the neighborhood, agents communicate in order to acquire data and coordinate actions.

An alternative is to **let agents self-organize** into green waves without communication. Until some years ago, non-communication between traffic lights and local interaction was well-established due to the unattractive costs of communication.

Swarm intelligence is one approach for coordination without communication that is realized by `de Oliveira et al.` [75]. Here, experiments show that coordination of the agents is done without centralized management. The key is that in open, dynamic environments, agents need to adapt to changing organizational goals, to manage their interactions with other agents, and to deal with the availability of resources. This relates to models of learning and adaptation. Nature inspires multi-agent systems with observations from social insects. Therefore, `de Oliveira et al.` [75] models intersection controllers like social insects, where signal plans are tasks. The vehicles produce pheromones throughout their paths, and more intensely when stopping at a traffic light, which is observed and evaluated by the intersection agent. This can trigger a change in signal plans without any centralized control or task allocation mechanism. However, that approach implies some time in order for the observation and evaluation to converge to a stable coordination. Especially in highly dynamic environments this is a negative aspect.

The framework of Organic Computing [383] is also applied to traffic. Prominent examples are the works of `Prothmann et al.` [280], [278] and `Tomforde` [346] for adapting complex and self-organizing systems. In `Prothmann et al.` [280], an Observer/Controller system for optimizing the intersection is introduced, as well as a self-organizing coordination mechanism that allows traffic-responsive progressive signal systems or green waves. The traffic light controller investigates the system and the observer monitors the local traffic demand and evaluates the performance of the actual signal plan. The observer selects and optimizes traffic light plans with a two-level learning mechanism: a learning classifier system and optimization by evolutionary algorithms. The approach is tested in isolated intersections, progressive systems and self-organized routing.

The above mentioned approaches focus on traffic control strategies and especially signal optimization without explicitly considering the demand. **Learning** agents or systems control their local environments and coordinate their actions and therefore try to balance the demand and control sides.

There are variations of learning agents depending on how they control their environment and how they coordinate their actions. At one intersection there are traffic lights and vehicles. Each affects the other based on the waiting time of the vehicles.

Reinforcement learning is used by `Wiering` [378] for the traffic light agents to minimize the overall waiting time of vehicles in a grid. Expected waiting times of vehicles are estimated by a value function, which the traffic light agents learn for different settings and then compute policies. The same value function and the policies calculated by the traffic lights are also learned by the vehicles in order for them to select the optimal routes to their respective destinations. This can be considered as co-learning.

Another method of reinforcement learning by `Steingröver` [327] describes a global control perspective with local actions for minimizing the total travel time of all vehicles in the network. The learning task is vehicle based such that the state representation of waiting times for individual vehicles is aggregated over all vehicles around the traffic light in the intersections which are represented as agents. As in `Wiering` [378], it is assumed that the traffic lights are able to know the vehicles' value functions and have additional information like their destination. The state space gets bigger the more information is known about the single vehicle [see `Bazzan and Klügl` [29]]. Investigation is done for other state representation forms with different learning abilities, generalization and performance.

The Q-learning method is investigated in `Bazzan et al.` [30] as one strategy of traffic lights, others are to keep the default signal plan or to run the green time longer for more throughput in a greedy fashion. Also, drivers of vehicles have three strategies: the randomized selection from among the available routes, selecting the route with the best average travel time greedily or adaptively choosing the route with the best probability by calculating average travel times needed. Therefore, the drivers select from large sets of routes and the control is done by the decentralized traffic lights. The objective of the drivers is the minimization of travel times whereas traffic control is done locally to aim for the minimization of spatial density like waiting times around the traffic lights. This work is presented in a typical commuting scenario modeled as a grid with many origin-destination matrices for the drivers who select their routes.

This method was extended to include on-the-fly re-routing by `Bazzan and Klügl` [28] where drivers avoid jammed intersections and adapt according to their perceptions. Thus, re-routing can compensate for inefficient traffic control. However, reinforcement learning by drivers is not included in this approach.

`DesJardins et al.` [83] focus on **collaborative driving** with an agent-based cooperative architecture to control and coordinate vehicles. It is a multi-layered architecture with an action and a coordination layer. The former manages low-level vehicle actions like braking, accelerating or steering and the latter integrates cooperative decision making between the vehicles on the high-level action. Both layers use reinforcement learning for handling complex details of vehicle control and different interactions between vehicles automatically. Therefore, this approach shows efficient results for vehicle control and coordination.

3.4.2 Decentralized Architecture

Cooperation of agents exists in Teamwork by `Tambe` [335] and Joint Intentions by `Jennings` [187].

MAS considers methods of coordination and cooperation e.g., in [71] [103, 302]. For an abstract simplified domain without considering communication and sensory aspects `Barrett et al.` [22] examine models for ad-hoc agent teamwork. There are various multi-agent grouping technologies applying general coordination and cooperation processes, e.g.,[68], [88, 234] [379], but no solutions tailored to the traffic domain.

A few projects bridge the gap between natural and artificial group behavior (works by JL Deneubourg like `Sueur et al.` [331]). They define some key terms for swarm intelligence and work with physical and simulated ant colonies or a human world and an artificial simulation. Furthermore, they discuss issues of relating the local and global behavior of a distributed system, as well as central and decentralized networks and shared and unshared aspects of decision making.

In robotics research there is a shift away from theoretical and simulated work towards physical implementation. Most focus on the control of a single agent, but several research works have obtained knowledge on and experimented with multiple physical robots. Simulations of group behavior in situated systems are becoming more common, like RoMADS: Robotic MultiAgent Development System and little Lego mindstorms control robots.

From the field of robotics there is a good approach for describing basic behaviors inspired by biology [245] as ubiquitous general building blocks: avoidance, safe-wandering, following, aggregation, dispersion and homing including information exchange and cooperation. Group behaviors in the spatial domain are goal-driven spatial-temporal patterns of agent activity. Spatially fixed organizations of agents correspond to achievement goals, whereas many spatio-temporal patterns correlate to maintenance goals. All behaviors have optimized interaction dynamics based on conserving energy and maximizing interaction or synergy within the group.

'Adaptive group behavior' by `Mataric` [245] is very comprehensive for social groups and makes use of basic behaviors as building blocks of adaptive agent control, social interaction and learning. Mataric distinguishes spatially fixed organizations, which want to achieve goals, versus spatio-temporal patterns, which have to maintain goals. In future, a similar concept could be applied to infrastructure, which wants to create groups, versus vehicles, which want to maintain the groups. `Mataric` proposes three optimization criteria for interaction dynamics: 1) conserving energy 2) maximizing interaction 3) maximizing synergy within the group. A variety of spacial interactions and tasks for a group of mobile agents are covered by Safe-wandering, Following, Dispersion, Aggregation and Homing as the basic set for the spatial domain. Thus, the model can do high-level composite behavior for achieving a variety of individual and collective goals such as flocking and foraging.

`Dresner and Stone` [89] consider reservation-based intersection control with an intersection manager (IM) instead of conventional traffic lights. The autonomous vehicles inform the IM of their arrival time, velocity, direction and other properties like their acceleration. The research requirements prohibit vehicles from turning and lane changing and assume constant velocity. The intersection manager simulates the vehicles' routes and, in case of a booked time slot, it rejects the new reservation request. That requires the vehicles to slow down and try to make a reservation for the next available time slot. When the request is accepted by the IM, then it must be guaranteed or, if the autonomous vehicle is not able to meet the reservation, it can cancel it itself.

In continuing work of `Dresner and Stone` [90, 93] the research requirements were modified so that reservations are necessary for vehicles to enter the intersection, speed changes are allowed so vehicles can decelerate after a denied request, but turning was still not allowed. That means that the vehicles need to follow the advice they receive from the IM and they cannot improve their routes if they already have a granted reservation to avoid last-minute changes by the driver. The authors did those modifications to open the system up to mixed traffic: human drivers and autonomous vehicles. Yet, it is a major challenge to cope with mixed traffic and to ensure driver behavior for an overall system balance when, on the other hand, autonomous driving should be possible.

In `Dresner and Stone` [91] they extended their work to a **market-based** approach to improve the delays and deadlocks from the drivers' reservations that result because the IM processes requests on a first-come, first-served (FIFO) basis.

With the focus on a single intersection and the different valuations of reduced waiting times of the drivers `Schepperle and Böhm` [307, 308] propose agent-based traffic control with auctions. With this metaphor, a priority of drivers can be modeled at an intersection to increase the overall driver satisfaction. The procedure is as follows: the intersection is contacted by the vehicle to receive an initial time slot for crossing. The vehicles can negotiate an improved time slot for crossing the intersection with each other. Later arriving vehicles can get time slots which have already been auctioned off. The vehicle agents adapt the speed depending on their time slot. In `Schepperle and Böhm` [308], the authors discuss the challenges of realizing such a system and present ideas for making valuation-aware intersection-control mechanisms operational.

An hybrid metaphor for the problem of congestion by `Tumer et al.` [352] includes the idea of minority games where vehicles aim to maximize their personal objective function. Here, the difference between actual and desired arrival time and the global system seek to minimize system-wide delays from the view of a city manager. Thus, the global and individual perspective is taken into account. The authors ensure that the agents get utilities that are good for the system level behavior because the worst outcome for both is the greedy choice for their own best interest. Therefore, the utilities need to be balanced. The drivers need to be aligned with the system utility, meaning that, if the agents maximize their own they also need to improve the global utility. On the other hand, the global system needs to be sensitive to the agents' actions in such a way that the appropriate execution option will be selected without taking all agents' utilities into account. Thus, multi-agent learning algorithms are applied for vehicles and the system to select the departure time in such a way that delays and jams are avoided at certain times of day.

This research was extended by `Tumer et al.` [351] for the selection of the departure time and lane choice influenced by the agent reward functions. This is especially interesting when the city manager's reward function is not aligned with the social welfare function. Further investigation needs to be done on the incentives of the agents so that two objectives can be maximized for the global and local equilibrium. Therefore, it must be ensured that the system's advice is followed by the vehicles. Possibly electronic institutions can contribute here when the vehicles are autonomous, since incentives and rewards can be communicated and enforced. Human drivers have no obvious rewards other than i.e., travel times or fuel consumption. But travel times or economic driving by themselves are not a good measure for system level utility. That means that how to model and simulate fitting mechanisms so that drivers' utilities are aligned with the system reward and the system is sensitive to reward changes of each agent is an open research problem.

A cooperative car navigation system with route information sharing is proposed by `Yamashita and Kurumatani` [385]. In their work, each vehicle transmits its current position, destination and route plan to a route information server. There, future traffic congestion is estimated based on the actual traffic information and then those estimations are sent back to the vehicles which then re-plan their routes. For evaluation, the individual incentive and the social acceptability are used to show the effect of the route information sharing system. The authors show that experiments generally satisfy both, and the effect of improving traffic efficiency is given.

3.5 Group Models

Specialized literature on platoons, communication and algorithms are listed in the following items:

- platoons: overview of platooning systems [38], challenges of platooning [37] [117], vehicle sorting for platoon formation [151], platoons of vehicles viewed as MAS [152], platoons on public motorways [294], analysis and control of platoons [361]
- communication: Distributed Multi-vehicle Cooperative Control [290] and group formation in VANET's [56]
- algorithms for team cooperation in a multi-vehicle unmanned system network [316]

A theoretical model of autonomous vehicle groups is presented in Chapter 5.

Both horizontal and vertical dimension of cooperation are presented in Figure 3.2 and examined in order to note similarities and differences between them.

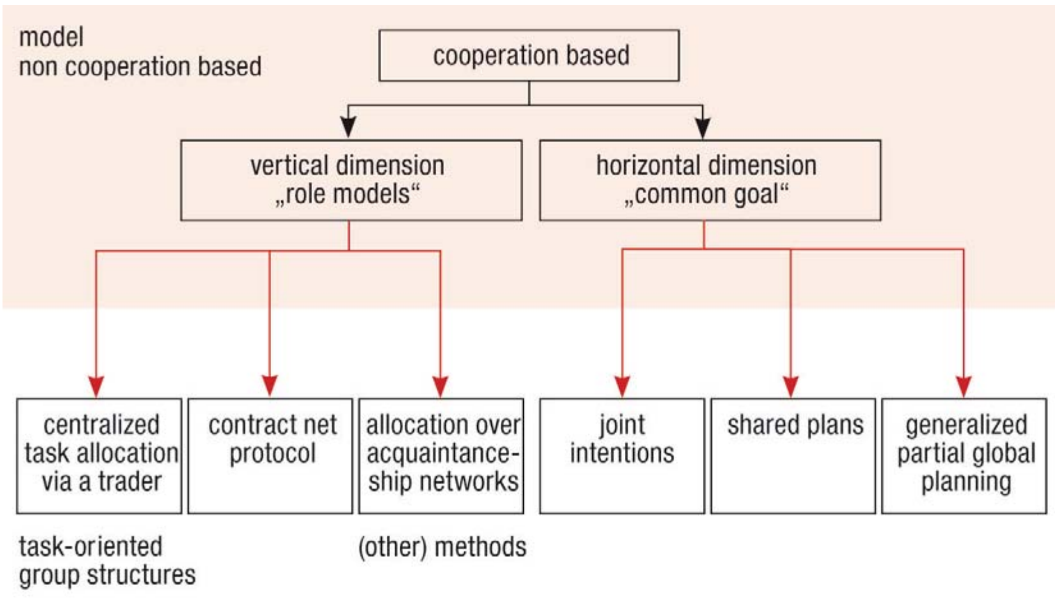

model
non cooperation based
cooperation based
vertical dimension „role models“
horizontal dimension „common goal“
centralized task allocation via a trader
contract net protocol
allocation over acquaintance-ship networks
joint intentions
shared plans
generalized partial global planning
task-oriented group structures
(other) methods

Figure 3.2: Different Group Forming Techniques (cf. [230] p. 37).

In the following, the definition, analysis and comparison of the group work models with regard to structural, functional and organizational aspects are presented.

In the horizontal dimension, group models involve Joint Intentions, Shared Plans and Generalized Partial Global Planning, which are assumed to be ideal for approaching autonomous vehicle group formation.

3.5.1 Group Norms

A norm is a behavior expectation originating from values and beliefs which all members of a group share (cf. [306] p. 22). Groups can discuss and agree on norms which guide their group and hold one another responsible to them, finding inconsistencies within the group. The basic rules of a group are a set of group norms based on the core values and beliefs.

Relative to the intended goal and according to the priority or utility level of the rule, a rule is chosen. These priorities are either defined a priori, or computed from heuristics, or inferred from the last use of the rule. This means that the rule's priority increases, according to its capacity to further the agent's goal, each time the agent uses it to reach its goal.

Norms emerge in groups that encourage high or low effort usually as a reaction to coping with the group task and work situation.

Metanorms [10] ([11] Complexity of Cooperation Chapter 3 Promoting Norms) can promote and sustain cooperation in a population. Other mechanisms for the support of norms are also important. These include dominance, internalization, deterrence, social proof, membership, law, and reputation.

Decentralized formation of groups within a multi-agent organized society are described in the following. Organized society means institutions which use certain norms. According to `Villatoro and Sabater-Mir` [365] (p. 1)

> a group is defined based on the set of social norms used by its members: all the agents using the same set of norms belong to the same social group.

They introduce different mechanisms for recognizing others as members of a specific social group by using three algorithms:

- the whitelisting algorithm, which works as a recommender of trusted neighbors,
- the blacklisting algorithm, whose basic function is to defame the non-related agents inside a certain social group,
- the labeling algorithm, which publishes information on the interactions with different agents, allowing the rest to access that information.

3.5.2 Goal-based Models

This section compares group work models in the **horizontal** dimension, considering the respective theories and rule identification or integration, understanding of interaction, the ability to cooperate and communicate between agents both at and outside the workplace or planning and controlling development processes.

In horizontal cooperation there is no hierarchical structure. All of the agents follow the same goal and are capable of the organizational cooperation necessary to find and execute the plan together. This is a structure where the inclusion of all agents in the decision-making process and specifically the choice of the method for exchanging information and potential interaction is essential. To one extreme extent, no superior agent exists and each of the agents fulfills tasks. This, in particular, may lead to competition between agents.

The flat structure in horizontal cooperation may appear to be a drawback for the specification of a solution algorithm. This dimension emphasizes the importance of social interactions in social rationality. But it also motivates the notion of bounded rationality, because of possible difficulties such as unstable resources, uncertain information, etc. To achieve the individual goal, each agent takes into consideration the limited resources available. An advantage of the horizontal cooperation is the negotiating skill in decision-making processes, combining the effectiveness of both cooperation and competition behavior in group work.

According to `Tambe` ([335] p.84), group work theories such as Joint Intentions, Shared Plans or Generalized Partial Global Planning are not relevant for direct implementation, but in agent modeling. RETSINA, STEAM and TÆMS are some examples of implemented general models of groups each based on shared plans, joint intentions and Generalized Partial Global Planning. Practical applications have been initiated, but success in making these models useful in general is limited, due to helpful explanation strategies or planning. The difficulty of effective model implementation in a dynamic domain requires additional requirements. Several models of group work intend to make it easier to deal with the difficulties, uncertainties or the inflexibility in coordination and communication of agents. Through reuse of group work capabilities, the implementation effort can be reduce and the consistency in group work ensured. Thus, non-coordination is avoided, resulting in improved flexibility. Considering and combining the group theories, several models can be developed.

In the following paragraphs, three general MAS group formation methods are described: Combinatorial Auction 3.5.2, Communication-based Group Formation 3.5.2, and Lose Teams - Groups 3.5.2, followed by different methods of the horizontal dimension, which describe the structure of goal-based groups in Multi-agent systems (MAS). Three standard group methods are also depicted:

- Joint Intentions: STEAM, RETSINA 3.5.2
- Shared Plans: RETSINA, STEAM, COLLAGEN 3.5.2
- Generalized Partial Global Planning (GPGP): TÆMS 3.5.2

Combinatorial Auction

This subsection presents a group decision-making mechanism based on a combinatorial auction [119, 288, 305] that agents can use to solve instances of the Initial-Commitment Decision Problem (ICDP). In this mechanism, called the ICDP mechanism, agents bid on sets of tasks in the proposed activity. Of central importance to the mechanism is that agents can make their bids dependent on temporal constraints, thereby enabling each agent to protect its private schedule of preexisting commitments. In addition to solving instances of the ICDP, the ICDP mechanism can also be used to solve task-allocation problems in the context of multi-agent activities to which the group is already committed.

The combinatorial auction can be seen as an optimal way for selecting group members. According to `Hunsberger` ([178] p. 34), members of agent groups are utility-maximizers as described in the following quote.

> When rational, autonomous agents encounter an opportunity to collaborate on some group activity, they must decide, both individually and collectively, whether to commit to doing that activity. We refer to this problem as the Initial-Commitment Decision Problem. It is assumed that new opportunities for collaborative action arise in context: each agent may have pre-existing commitments to other individual and group activities. It is further assumed that agents are utility-maximizers and, thus, will only commit to group activities if their individual net profit is expected to be non-negative.

The ICDP mechanism consists of five steps:

1. The cost of each intention is given first from each agent.
2. It is compared with all other intentions received from other agents.
3. It is decided if it can be achieved at the same time or not

4. The agent decides whether he will be able to fulfill the goal within the time borders set for him.

The best combination still has not been calculated. It will be used for the best requirements, which are chosen in this step.

Real-Time Group Formation

Another variation is Real-Time Communication-based Group Formation, which defines the group plan as an overlap of all plans including all agents. Each agent is characterized by the atomic actions. These individual skills to fulfill different actions give the Real-Time Group Formation model a hierarchical structure, where the group leader stays at the top of the hierarchy and has all the information about the plans of all the agents and uses them to build a group plan. After asking all the agents for their skills, the group leader decides to form a group using all the skills needed to fulfill the plan. Agents do not take on additional tasks and comply with the requirements.

Lose Groups

`Cohen and Levesque` [68] propose another model without a hierarchical structure defined as a group, where the agents want to share their own intentions with the others, called Belief-Desire-Intention agents (BDI). The group is formed not by a leader as in Real-Time Group Formation or a central authority as in the combinatorial auction method, but rather through the intentions of members.

Joint Intention

Joint Intentions ([382] p. 223, [315], [229], [188] [189], [190]) is an agent group model that works analogically to human group models. Nowadays, researchers insist on a more comprehensive concept of building agents with joint intentions in accordance with the concept of practical reasoning processes. This is attributed both to the positive influence in social interaction through stability and predictability, and also to the necessary flexibility and reactivity beyond the need to adapt to new environments. According to `Wooldridge` [382], two types of coordinated actions are recognized: cooperative and non-cooperative. There is a significant difference between both coordinated actions with regard to reasoning and organization method. In order to recognize this difference, the example of a country's foreign exchange reserves is helpful. Some countries' oil resources have enabled the country to reach better income status, with a modern infrastructure, and good education and health care systems, while exporting oil and raising their economical resources. At the same time, the foreign demand for this good is satisfied. This can be a cooperative action; in this case, that each country intends to exchange some goods as a consequence of the overall aim of overcoming the lack of the necessary resource. A counter example of a coordinated but non-cooperative action consists in the action of people running out of a house in the case of an earthquake. During the earthquake, some actions have been considered intensively and others have been ignored. The fact that the method of running outside is used is a result of considering their own safety intensively. Each individual has their own intention and ignores the goals of the others.

Joint intentions can be summarized as agents setting their intentions as their goal. An important issue is how the joint intention is achieved:

There exists a common goal p, which symbolizes the fulfillment of a joint action. The following requirements need to be fulfilled such as p will be accepted by all group members as a reachable goal:

1. All believe at the same time that p is not fulfilled at present.
2. All set p as a common goal, therefore p is the group goal.
3. All believe that p is a weak goal for all until it is fulfilled, unfulfillable or irrelevant.

Every group member has the task to pursue the goal p with all his possibilities. If one group member notices that p is fulfilled, unfulfillable or irrelevant, it is his task to tell the others. After the agreement that it exists as a joint intention, each agent can associate its own intentions to it ([326]).

Joint Intentions specifications are the joint mental state of a group (θ),the composition of which is to be set out in a joint agreement between the members concerned in achieving the same intention and committed to establishing a mutual belief (MB) for completing a group action.

Joint Intentions Symbolic

JPG $(\theta; p; q)$: JOINT PERSISTENT GOAL

θ : group;

p: group action completed;

q: Irrelevance clause;

WAG $(\mu; p; \theta; q)$: WEAK ACHIEVEMENT GOAL

μ: group member;

PWAG $(v_i; p; \theta)$: PERSISTENT WEAK ACHIEVEMENT GOAL

v_i: group members list;

REQUEST $(\mu; \theta; p)$: a request executed from μ to adopt the PWAG;

ATTEMPT $(\mu; \Phi; \Psi)$: an attempt to inform others of the PWAG of v_i to achieve p;

Φ: ultimate goal;

Ψ: achieving MB;

This reasoning leads to the JOINT INTENTION theory as a MENTAL-INTERCONNECTED group framework. While recognizing the differences between individual intentions and collective ones, `Levesque et al.` [229] concluded that increasing attention must be paid to issues of cooperative social RESPONSIBILITY as an important part of a group's identity. Another concept to which `Levesque et al.` contributed is COMMITMENT, particularly in groups of agents that should have not only the joint commitment to the joint aim, but also the individual commitment to an assignment. According to the status of the goal, a social CONVENTION must also be used, focused on specifying requirements for an impossible or achieved goal, as well as for a goal which is no longer present to achieve. This means that the agent must inform his group whenever it notices that it cannot achieve the assignment and decides to abandon his goal. A type of COOPERATION where the agent achieves a JOINT COMMITMENT for the goal p is called JOINT PERSISTENT GOAL (JPG). Example:

- p: move the picture.
- motivation: Angela wants the picture to be moved.

Three conditions for satisfaction of JPG $(\theta; p; q)$ are:

1. There exists a Mutual Belief, that the goal p is currently not achieved: p = FALSE.
2. The only common intention of group members should be focused on achieving the common goal p.
3. There must exist a Mutual Belief that the achievement of the goal p consists of a Weak Goal (WAG), till the state of achievable, unachievable or irrelevant is satisfied.

WAG$(\mu; p; \theta; q)$ achievement specifies the following conditions:

1. $p(\mu)$ = TRUE: The group member (μ) intends to change the status from FALSE to TRUE.
2. MB(θ) = Pb(θ) : After comparing the status of p against True or False, μ must share it to the group, which adopt it as a Mutual Belief.

Such an approach to μ action would at first be concerned with weak goals capable of achievement in the short term, with the aim of attaining a mutual belief in termination of p through JOINT COMMITMENT. This step gives the group the possibility of having the current evaluation of the work situation, avoiding delays, misunderstanding or conflicts. The key to success here is the contact between group members, who stay committed to a course of action, find the Mutual Belief and use REQUEST-CONFIRM-CLAUSE conversation. This way of estimating commitment to the interest of the common goal of the group as a whole rather than to the interests of their own intentions is called 'establish commitment protocol'. Furthermore, a PWAG is a key to success within this protocol. PWAG$(v_i; p; \theta)$ syntax includes three steps:

1. REQUEST $(\mu; \theta; p)$ = ATTEMPT$(\mu; \Phi; \Psi)$. After sending a Request, μ has to achieve p together with the other members and expects that each v_i adopt PWAG $(v_i; p; \theta)$.
2. In the second step, the members of the group provide an answer, respectively confirming or refusing the status of the member as v_i having to achieve PWAG.
3. Only when the whole group has given confirmation is this JPG$(\theta; p; q)$ generated.

According to `Searle` [315], the treatment of Joint Intention implies the main question about the origin of the individual intentions, considering the fact that

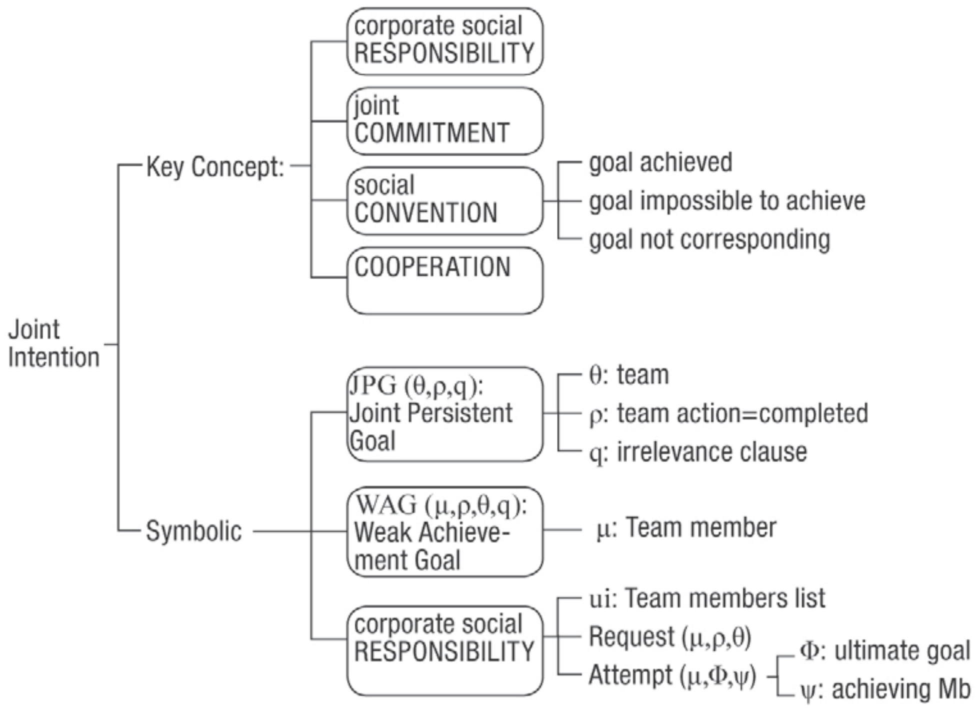


Figure 3.3: Joint Intention Model.

joint intentions would not summarize the individual ones, while being executed by individuals. Conventions made by the individuals incorporated herein have been individually accepted. Whether or not it pays to abandon the goal because of individual inspection has to be decided upon by the individual.

STEAM (simply, a Shell for TEAMwork) is a hybrid implemented general model of teamwork based on Joint Intentions theory and partially also on Shared Plans theory (described in the next Subsection), with the Soar Cognitive architecture and the following hierarchical structure:

- joint intentions
- individual intentions
- beliefs about others' intentions

STEAM operates in three different types of teams which include more than two or three agents and have a complex organizational structure of team-subitem hierarchies. It pays practical attention to reducing the costs of communication. Due to this structure, there are some noticeable benefits of STEAM:

- Flexible teamwork available through Joint intentions communication and 'Soar' architectural mechanisms [261].
- Explicit description of reactive team goals and plans supported at a significantly higher performance level of the team.
- Recognition and specification of new responsibility and assignments in case of failures.

- Reorganization: Failure recovery.

The information on teammates' activities regarding the recipe, which describes the action to be performed, is limited, subject to recipe existence, while the details of the recipe are missing. The agents do not need to recognize the detailed plan or track of the teammates, but only the joint intention in order to attend to the execution of the next step and stay updated on the status of the team. In the domain Attack, for example, there is also a high-level indication such as role execution-state available, but not in ROBOCOP domain. In STEAM, domain-independent mechanisms are incorporated in order to successfully realize the role-monitoring and performance state strategy. This is also applied in impossibility of performance, when all of the team members failed to perform their goal, and in determining a new plan which appears reasonable according to the circumstances. The repair and re-planning process begins with analyzing the failure. It is only possible in case of a critical role failure through team reconfiguration; when all roles fail or in the case of no possible substitution it is irreparable. STEAM is applied in three domains:

- Attack
- Transport
- RoboCup

For the implementation of a domain, new requirements can include both generalizations in case of special case operators or modification without adding new coordination plans in case of development.

Shared Plans

Shared plans - in short - are 'intending-that' expectations that are directed to another collaborative agent [146].

There are Full Shared Plans (FSP) and Partial Shared Plans (PSP). A FSP has the following attributes:

1. All members of a group GR believe, at the same time, that they can fulfill a joint operation within a time T_a.
2. All group members of GR believe, at the same time, that R_a is the recipe for the execution .
3. For every step b_i of R_a is set a subgroup GR_k which has a FSP for b_i with the recipe R_{b_i}. Other members of GR believe in the existence of a recipe such that GR_k can fulfill the task b_i and a FSP exists (but they do not need to know the recipe themselves). Other members of GR expect that (intending-that) GR_k can fulfill the task b_i with the help of whatever recipe.

In reality this will never be. There are three reasons for that:

1. The recipes are just partially specified.
2. The task allocation is not agreed on.

3. Individuals and subgroups have not agreed on a joint intention.

Therefore, reality just has PSP, which are snap-shots of the system, and agreement processes are possible.

Shared Plans - explained in detail - is an intentional-attitude based model, where 'the intention that' should be attainable in its collaborative or group joint-action nature, by identifying individual actions referring to the current state (collaborative actions as well) according to a set of axioms to improve the performance of the subgroup and group (sometimes also referred to as 'team'). `Tambe` [335] has distinguished between two types of Shared Plans:

- Full Shared Plan (FSP),
- Partial Shared Plan (PSP).

FSP Symbolic **Recipe:** $R(\alpha)$ **FSP**(**P**, **GR**,$\alpha, T_p, T(\alpha), R(\alpha)$): The full shared plan P of a group GR at time T_p, trying to do an action α at time $T(\alpha)$ by executing a recipe $R(\alpha)$ **P:** plan **GR:** group α**:** action T_p**:** time for completing the plan $T(\alpha)$**:** time for performing the action

Shared Plans Specifications FSP(α): The method of work in FSP is determined by precisely indicating the *full* specifications of a joint activity through a complete recipe R(α), to be executed under specified constraints. FSP (P, GR, $\alpha, T_p, T(\alpha), R(\alpha)$) holds when the following conditions are satisfied:

1. All members of group GR share the mutual belief in the DO statement for executing the action (α) at the same time $T(\alpha)$.
2. All members of group GR share the mutual belief in the recipe to be used, recognizing its validity for (α).
3. For the set of actions β_i the following can be emphasized:
   (a) Each individual has an analogue FULL INDIVIDUAL PLAN. In this case, the symbols have the indices k and i respectively : GR_k for the subgroup and R_{β_i} for the recipe.
   (b) The rest of the group GR mutually believe in the existence of the recipe to make the fulfillment of the action β_i possible, but does not have specific information.
   (c) The rest of the group GR mutually believe that GR_k can fulfill β_i with the help of some recipes.

The Shared Plan Theory tends to explicitly describe the entire content of all the intentions and beliefs of a group. That is why the FSP is regarded as a limited case. PSP expresses a partially precise view of the group's mental state depending on the situation, which means that it does not only consider defined information, but COMMUNICATION and PLANNING can also be introduced. Full Shared Plans theory is not treated further for the following reasons:

1. Forming a Shared Plan for each upcoming step (β_i) can be considered a partially evolved process, particularly regarding the recipes over time.

2. Partial task specification consists of the debate over the tasks which no agent intends to deal with, and which still remain to be determined by the agents' decision to contribute or convince other agents to cooperate.
3. Partial communication arising in particular as a result of intending that attitude with respect to both the group goal and any other agent's activities.

Receiving and responding to adequate or relevant communication from other group members should be taken into consideration, in favor of the improvement and generalization of communication capabilities (STEAM example).

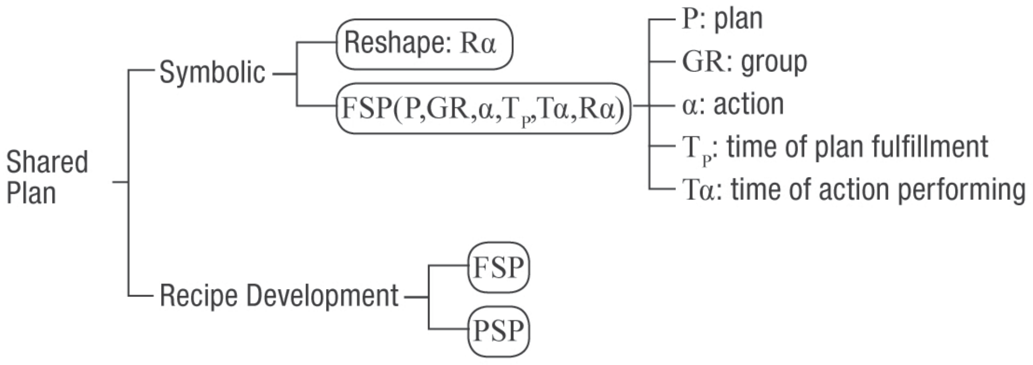


Figure 3.4: Shared Plan Model.

COLLAGEN [145], [146], [144] is a collaborative interface agent constructional method, based on the Shared Plans theory, applied in complex systems (complex or untreated assignment for agents with a collaborative interface) such as air-travel arrangements. It employs generation and interpretation algorithms, based on 'intend that' attitude reasoning and includes intentional, attentional and linguistic components each implemented respectively by plan trees, a focus stack and by an artificial negotiation language.

RETSINA [130] is another model of groups, in which both joint intentions and shared plans are combined and implemented. The following ones are considered to be new points of emphasis in the coordination of the group:

- social order and authority recognition, proposing and admitting based on the 'role' concept (see the following Section 3.5.4).
- Commitment check-points mechanism in order to improve communication based on the request-confirm speech.
- strongly oriented towards individual specification of the subgoal
- any group member can decide to give up a subgoal or goal, but it cannot leave the joint goal of the group.

RETSINA has been applied through group operating in a simulated environment.

Generalized Partial Global Planning

The brief overview of Generalized Partial Global Planning according to Lesser [226]: In general, every agent plans its own future activities listed in the following and takes the interdependency of activities of other agents into account.

1. A list of its goals /abstract tasks.
2. A tree structure depending on the subgoal with discrete probability distribution of the three dimensions quality, cost and time.
3. A precise quantitative rating of each task as to how it correlates with the superior goal of the agent based on solution quality, cost and/or time.
4. An allocation of the activities to the resources.

The agreements between the agents have three categories. Deadline agreements give a predefined end to the task, according to when it needs to be successfully fulfilled. Earliest start time defines the start time for task, and do agreements are time-independent [326].

Generalized Partial Global Planning theory (GPGP) - in detail - [228] [94] [335] [79] [80], [81] [226] [78] is a successful domain-independent framework based on multi-agent distributed planning, affecting general or specific situations and applicable to local environment-centered coordination mechanisms or large agent organizations. It has a decisive influence on the increase of the possibilities of efficiency and applicability through integration in more applications. Distinction between distributed and centralized multi-agent planning constitutes an important aspect of task planning and scheduling techniques used in GPGP. Centralized multi-agent planning is based on the concept of CTPS (Central Task Planner and Scheduler), which plays a central role in regulating the coordination and has also been tackled in defining the following requirements:

- specification: goal setting,
- planning: task decomposition into subgoals (programs of action) and
- scheduling: task assignments, agenda planning. Each agent receives the assignment according to their position, charge and competency.

It is possible for 'situation-specific' cases to be performed with concentration on a specific coordination with the new strategy of using basic knowledge instead of the dynamically gained and established set of knowledge, reducing the complexity of treating them. This can be seen as a good example of how more top-down and contracting types of coordination mechanisms can be successfully implemented, increasing the compatibility of multi-layered control architecture with an organizational design that identifies the coordination strategy. GPGP property is:

- contains a Coordination framework (extendable family of coordination mechanisms);
- uses GPG algorithm extension and generalization;
- is based on a system of cooperating agents acting according to the needs of a task environment;

- does not apply the collaboration theory;
- is expanding in the direction of domain independence;

According to Decker and Lesser ([81] p. 1) the GPGP approach summarized as:

> The GPGP approach consists of an extendable set of modular coordination mechanisms, any subset or all of which can be used in response to a particular task environment. Each mechanism is defined using our formal framework for expressing coordination problems.

There are several unique features in this approach such as:

- it is open for multiple domains;
- in the current task environment each mechanism is a response to given features;
- for different task environments appropriate combinations of mechanisms are adjusted;
- GPGP makes use of the agent's existing local planner or scheduler.

TÆMS fields of application includes a broad application area (refer to Lesser et al.([227] p. 2): information gathering and management, distributed patient scheduling in hospitals [271], travel planning, dynamic supply chain management (SCM) [367], intelligent home automation, coordination of concurrent engineering activities, and aircraft service team coordination [366].

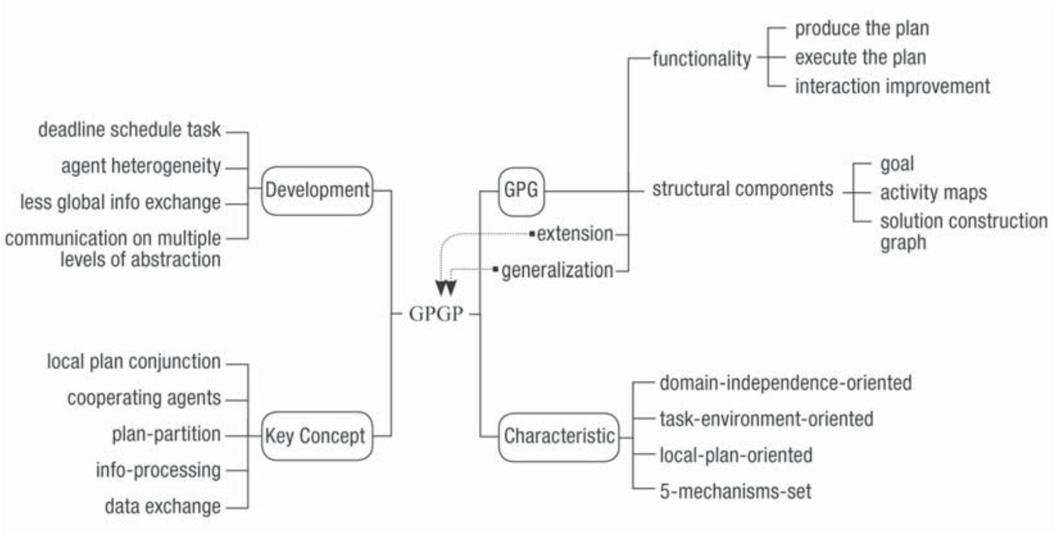


Figure 3.5: Generalized Partial Global Planning Model.

Additional regulations for the GPGP [80] concept have been issued for the improvement of the interaction between agents. In comparison to PGP, GPGP has the following properties:

- schedules tasks with deadlines

- allows agent heterogeneity
- exchanges less global information,
- communicates at multiple levels of abstraction.

The additional actions are the following:

- Updating non-local viewpoints;
- Communication results;
- Handling simple redundancy;
- Handling hard and/or soft coordination relationships.

All these properties can be seen as advantages to be used in different applications applying GPGP, making it more appreciated in multi-agent areas.

3.5.3 Comparison of Goal-based Theories

The analysis requirements in which the group theories are compared include:

- structural similarity (symbolic)
- team description
- agent architecture
- hierarchic structure
- complexity in communication and coordination and
- group size.

In the following comparison table 3.1 these requirements are presented.

Consider the requirements for a general comparison between the goal-based theories, which are related to the comparison table. Having analyzed the symbolic characteristics, some structural similarities are detected, such as the concept of the shared plan, collaborative group and intention-based action in FTP, which are analogous to Joint Persistent Goal, team and joint action in Joint Intentions. Joint Intention typically involves the PWAG concept, which generates behavioral constraints and improves the negotiations between the agents. Firstly, coherence is achieved by executing actions and rejecting the possibility for agents to cancel each other's efforts. Those situations are described for the blackboard example in `Wagner et al.` ([367] p. 119). Secondly, this concept allows the agents to observe the current state of the group work and helps in reducing the complexity of relationships between agents. This is a key concept for analysis, replanning and repair possibilities. Based on this analysis, especially of Shared Plans, changes concerning the time for performing the plan or the action or the necessary recipe for the execution of the Shared Plans are not found in Joint Intentions or GPGP.

The lack of hierarchical structure in horizontal group theories causes difficulties in finding solution algorithms. This is due to behavior of the agents which is hard to predict and, therefore, a solution strategy in case of conflict or lack of information, competence, resource or desire is difficult to develop. In Joint Intentions, the agent has to decide by itself whether it wants to drop the goal

Requirement Analysis			
Requirements	**Joint Intentions**	**Shared Plans**	**GPGP**
Symbolic characteristics	no time limits PWAG	T_α, T_p recipe	

Team Description	joint mental state;
	autonomous work;
	local view of activities;
	role-monitoring constraints;
	sub-group structure
		joint-action in attending;
		intentional attitude;
		sub-groups
			partially global view of activities;
			reactive to changes
			
			in the environment

Agent architecture	BDI model	BDI model	info asymmetry

Structure	no hierarchy (flat);
	explicit team;
	goal and plans
		joint goal and plan hierarchy;
		center based;
		intentional attitude based model
			no hierarchical structure

Communication	Request-Confirm-Clause
	conversation-commitment protocol;
	decision-theory not authentic;
	selective
		local commitment;
		
		axioms;
			load-balancing mechanism decision;
			
			local;
			non-local

Coordination	social convention;
	collaborative behavior;
	team-oriented
		complete or partial;
		recipe; collaborative behavior;
		Individual coordination mechanisms
			scheduler-task;
			structure
			
			other agent activity constraints

Interaction	multi-agent interaction	interaction between a human user; intelligent agent	multi-agent interaction
Group size	< 7 agents;	> 1 agents;	$5 - 9$ agents;

Table 3.1: Comparison of Goal-based Group Theories.

and conventions are made and accepted individually, having a flat structure in group organization.

In Joint Intention, the nature of an acceptable form of communication can be emphasized, which is based on commitment to attaining a mutual belief. The lack of authentic communication in Joint Intention is not to be considered as a disadvantage. In GPGP, the agents can receive commitments from other agents and expand their local view, improving coordination and coherence. In Shared Plans, the communication between agents includes submitting the necessary information regarding the mutual belief for the group to achieve an action.

On the other side, the decision theory communication represents a good solution, considering the communication cost and uncertainties or benefits.

Discussing the coordination of the joint mental attitude the recipe concept and the task structures respectively is considered for Joint Intentions, Shared Plans and GPGP. The task structure can be considered as a powerful structure characterized by the definition of the degree of achievement and can be customized in many specific situations.

Defining the optimal size plays an important role in group work methods, but it depends on many other group work specifications such as the nature of the task, the uncertainty of an event's threat, failure during task execution, deadlocks, unstable resources, continuous task enrichment, unexpected changes in environmental conditions, response time or increases in the amount of available information. Considering the coordination of groups, the optimal size in general should be between five and nine agents. But for each domain the degree of coordination has to be considered. According to `Tambe` [335]:

With more than six agents, the simulation with cautious groups could not be run in real time. Below and above this number, difficulties in making effective use of the theories are faced, that is, in managing the resources or reusage according to the environment and in obtaining accurate task scheduling, planning and reallocation respectively.

In GPGP, the concept of subgroups provides a possible solution for managing such difficulties. To grow effectively and communicate, the group has to maximize its size and conduct as many operations as possible within the T-time limits, which makes it harder to manage the problems with communication and coordination and to minimize the communication overhead. For years this subject has been discussed and several critical points have been dealt with in order to improve communication efficiency and group performance during a scale-up in group size. Two problems that could arise in terms of group stability and in communication and computational efficiency are:

1. The loss of physical and logical connection between agents and their peers.

2. Significant degradation in coordination; run time grows exponentially with the number of interactions between agents.

3.5.4 Role-based Models

For effective groups, members have clear, meaning agreed-on, roles, an know how to behave accordingly, including leadership. Thus, members can coordinate their actions to complete the task. With changes of tasks or members, the roles are clarified within the group, to counterbalance conflicts and stress.

Characterizing the leader role involves defining the relationship between the leader and other group members when it comes to handling group processes, structures, and functions. For more self-directed or decentralized groups, more elements of the leadership role become integrated into the roles of the members. Leadership is a broad term and often interpreted differently by each individual of the group, i.e., in terms of direct, unilateral behaviors like dictating a choice, or a more indirect role of influencing quietly behind the scenes like an omnipotent creator. Beliefs about leadership are embedded in group culture and norms, which helps groups set goals, solve problems, and communicate. Leadership can be clarified in terms of formal and informal leader roles and their expected appropriate behaviors.

The notion of roles is classically admitted for an agent [382]. This thesis considers a role as a set of (environmental and social) knowledge combined with a set of rules that allows it to reach its goals. Therefore, the definition of a role R is relatively similar to the agent definition.

A role contains some preliminary requirements in personal knowledge, in order to define dependencies between roles. There must be agents that will play it, and some consequences for the agents that embody it. A role also contains a workload for the agent that plays it, and agents have maximum workloads.

Role-based models support vertical cooperation with centralized multi-agent planning. Thus, cooperating agents within a group have a hierarchical structure. Classically, a superior agent covers the planning task including the task sharing for the subordinate agents. The superior agent decomposes the main problem into several subproblems, called task decomposition, and assigns them to the subordinate agents with different responsibilities. After completing their task, the agents communicate and send their feedback, which should be consistent with the corresponding expected result, called task reallocation.

Centralized Task Allocation

Centralized Task Allocation via a trader is considered a market-based approach and consists of three roles with their corresponding tasks illustrated in Figure 3.6:

1. Ordering party: Agents that delegate a specific task to another agent.
2. Service provider: Agents that are prepared to do the tasks of other agents.
3. Trader: One Agent that moderates between ordering party and service provider.

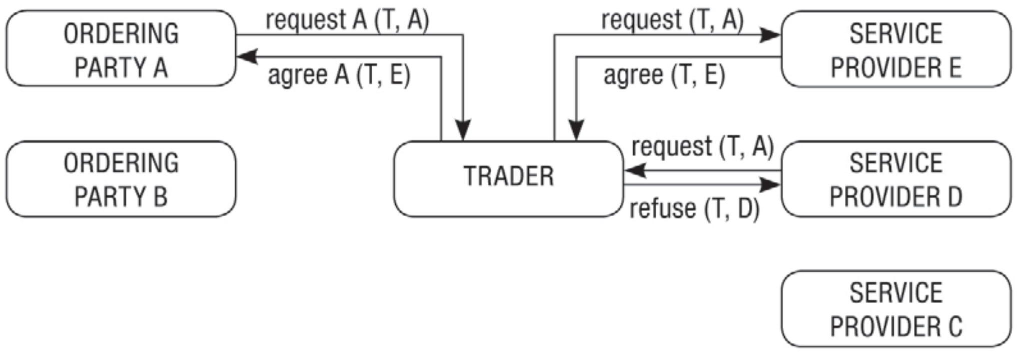


Figure 3.6: Centralized Task Allocation.

Contract Net Protocol

The contract net protocol (cp. [322] [324]), where the task decomposition is done between the requesting agent and the contract is established from the originator based on the feedback of agents' evaluation, could also serve as an example for vertical cooperation.

The Contract Net Protocol (CNP) can be used in multi-agent-systems for distributed problem solving. The problem is divided into smaller tasks on which agents can bid. The task is then assigned to the agent with the best ability to solve the task. This assures a good workload throughout the network. Tasks are distributed between agents by negotiating with each other. For this an initiator and a participant are needed. The initiator offers tasks on which the participants can bid during the negotiation phase. An agent can be either one.

In the propose state, the initiator asks all participants if they can fulfill a certain task. If the task is within their perimeters, they make a proposal to the initiator. Otherwise they reject the call for proposals. In the control state, the initiator checks all proposals and contracts the winner with a task. The winner is determined by a distance function. The other proposals are rejected.

To summarize, the following steps are negotiated:

1. RequestForBids(T, X) - request to a bidder X to show the offer concerning the task T.
2. Propose(T, O) - positive answer to the message RequestForBids. O is the offer for the task T.
3. NotInterested(T, Y) - refusal of the task T of the bidder Y.
4. Award(T, C, X) - Message of the manager to the bidder about the completion of the contract C with him.
5. Accept(T, Y) - verification of the bidder to fulfill the contract.
6. Refuse(T, Y) - refusal of the contract of a bidder Y.

Another variation of the CNP allows the bidder to break down the task received into subtasks and to offer to do those, that means being a manager himself (ref. [323], p. 359 ff., [177] p. 100 ff.).

Problems of implementation are categorizing the offer, addressing potential bidders and the existence of multiple managers.

There are special conditions in which no agents are currently available during a proposal phase. A broadcast to all agents should be avoided, because communication is expensive.

The extended Contract Net Protocol

In the original CNP, a contract is made by simple proposals or counter-proposals followed by an 'accept' or 'reject'. The extended CNP offers more interactions, so that agents can argue about the contract. For example, an agent can persuade another agent by offering him benefits. This includes threats, rewards or promises. However, the decision making process becomes much more complex.

To avoid large overhead, the communication must be decentralized. A central decision maker should not be required. This minimizes communication and data retrieval.

Allocation via Acquaintanceship Networks

Distributed allocation of tasks has no central instance. The positive aspect is that not everything is out of service during a system crash, whereas the negative aspect is that it is complicated to secure the coherence of the agents.

1. Direct allocation: assumption of direct knowledge of each trader
2. Allocation through delegation: forwarding of tasks to other agents.Therefore a connection without direct contacts is possible.

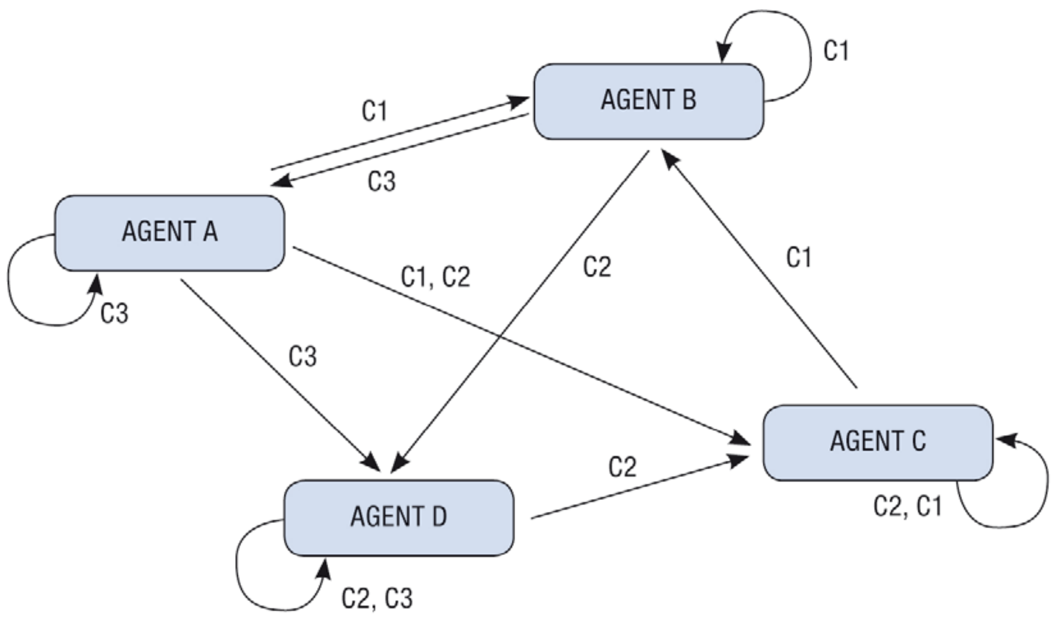


Figure 3.7: Allocation via Acquaintanceship Networks.

This is a simple method, but requires knowledge to identify a specialist. Additionally, it needs a mechanism to help establish indirect contacts. One agent needs to ask all the agents that it knows. It stops if somebody was found or the whole network was searched (graph search algorithm). Care must be taken with the simultaneous acceptance of one task by many agents.

3.5.5 Comparison of vertical dimensions

Summary of the group methods and structure in the vertical dimension: each method - task allocation via trader or acquaintanceship networks and CNP - has its pattern for task allocation and agent organization. The role model has universal validity because it has no fixed hierarchy. In the horizontal dimension, a definition of the common goal is enough to represent the group/team to the outside. The methods GPGP and Joint Intention realize their own methods for coordination and organization within the group. The difference is that there is no additional universal commonality other than the preconditions of a common goal. For example, an agent searches for a group that fulfills 'A' as a goal and, if he finds the group, he can directly ask it which methods are used to fulfill the goal.

3.6 Assumptions

This refers to Chapter 4, which describes the approach of the MATI simulation developed in this thesis, and Chapter 6, where the scenarios are described. The application area for the novel model of autonomous vehicle group formation is rush-hour in urban traffic. The validation of the criteria of efficiency needs to be evaluated. Therefore, the following measurements are planned:

- with and without groups,
- static and dynamic algorithm,
- homogeneous and heterogeneous vehicle types
- evaluating the metrics of travel time / throughput and stop time / delay.

The goal is to extend present to future technology while only considering autonomous vehicles in dense morning rush-hour traffic, which corresponds to thick traffic (LoS category C to D). The goal is to improve the capacities in rush hour, minimize traffic bottle necks, and avoid traffic jams and stop and go behavior. These challenges facing dynamic urban traffic can be solved through innovative technologies of autonomous agents, which are represented by vehicles for the purpose of this thesis in order to combine research of Dynamic Traffic Management, Data Mining Techniques, and V2X communication.

Each field has assumptions including the general settings of urban traffic, the simulation environment, the specific agent settings, and the notion of vehicle groups.

3.6.1 Settings of Urban Traffic

German standard streets for cities are assumed, not wide or narrow streets or bridges and tunnels. This means, right hand traffic with buildings on both sides, crossings and intersections with more than one lane for each direction. Lanes can open and close or are dedicated to the directions of driving. The time, position of origin and destination are influencing factors, therefore they are stochastically distributed and vehicles placed randomly in the networks. Collisions with other vehicles, plants, animals and buildings are excluded from the simulation through mathematical models which keep the gaps and avoid collisions. The speed is adopted from German standards for city traffic based on measurements made in Hanover with standard length and weight vehicles. The weight and speed of vehicles is a dynamic parameter which can influence the group behavior and is respected in the investigations. In order to compare characteristic differences, four different vehicle properties are investigated: small, big, fast, and normal. This is meant to cover the major vehicle categories like bus, limousine, city car and cross-country vehicles. Priority or specialized use like ambulances, police or fire service or horse carriages are out of scope. Two-wheel vehicles such as bikes, scooter, mopeds or e-bikes can be covered, whereas trams, ferries or boats and pedestrians are not covered by research.

The data was collected in connection with the Hanover trade fair for morning rush-hour and therefore represents event categories which could be also soccer games, demonstrations, concerts or accidents. The distance is within a small network of roughly ten intersections. As obstacles, only traffic lights are considered; others like construction, signs, cross-walks or service stations are topics of future research. Weather aspects and the condition of the streets are not investigated. Traffic jams are to be avoided by selecting a new route (by the parameters short, fast, plane). The coupling of vehicles is calculated based on origin and destination data correlating with their route preference and vehicle characteristics. Therefore, the groups are quite dynamic and vehicles are not forced to join a group.

3.6.2 Settings of Simulation

One major part is the traffic simulation which needs to be manipulable for agent extension, open for different scenario models including the mathematical models for realistic traffic dynamics, and measurable with analyzing tools. One gap is identified when listing the requirements for vehicle groups in urban traffic, namely, that there is no platform available which is sufficient for AVGF. This gap is closed in Chapter 4.

An environment is needed which provides similar flexibility to open-source environments, so that simulation entities are manipulable. Exchangeability is essential to make the tools future-proof and generalizable: interpreter, simulation environment, and analyzing tools should be exchangeable via a standard. This allows adaptation and manipulation of structural specifications, so that own algorithms can be implemented with the goal of bundling functionalities with existing standards. Environment and interpreter should not be coupled within the run time of the simulation, but beforehand. Traffic management is often static, having a centralized architecture. The goal of this thesis is to make it more dynamic using agents, which are, as well, often centralized but with a shift towards decentralization. The outcome using role-based hierarchical Contract Net Protocol (CNP) algorithms are hybrid groups; this fulfills the second identified gap. Other goal-based algorithms of Generalized partial global planning (GPGP), Joint Intentions (JI) and Shared Plans (SP) did not seem feasible for realistic traffic simulations.

Often traffic management is centralized with the focus on traffic lights as the most decentralized entities, but vehicles need to be influenceable for this thesis. Agent-based simulation systems often do not provide enough traffic dynamics. The MATI simulation model of urban traffic is a simplified version of real-world urban traffic similar to `Dresner and Stone` [89]) in their settings. The settings for this thesis are described in the following. Autonomous vehicles can accelerate and decelerate in order to go straight, left or right. Due to the microscopic traffic simulation, the vehicles enter and leave on sources and sinks, thus, they cannot park and they hardly ever turn due to mathematical driving models. Therefore, all vehicles begin traveling at roughly the same speed (due to the urban speed limits this is around 50 km/h), depending on the homogeneous or heterogeneous vehicle characteristics, especially on their desired speed, which can be up to 10% over the speed limit.

Urban traffic is rather complex. Thus, the model simplifies the analysis and implementation without reducing the fundamental challenges of the problem. The aim of this thesis is that vehicles drive to their destinations in a minimum time measured with travel and stop time. Thus, it is assumed that each vehicle tries to travel at their maximum restricted speed. Almost every vehicle in production is capable of attaining the speed limit of any street, so this assumption is not a far stretch. For simplicity, all vehicles of the same type are assumed to have the same desired velocity, which may depend on the lane. Additionally, vehicles form groups in order to improve their times. They use a smaller security gap while in group action.

3.6.3 Settings of Agents

The agents incorporate the BDI logic and have a social and cooperative behavior which includes communication. Additionally, the agents need the ability of coordination in order to have group effects. The coupling of agents with the entities in the simulation environment is essential. The agent communication needs to take place in real time. In general, the time and cost of communication are not manipulable, thus, communication needs to be minimized and substituted with interaction or other modes of interaction when possible.

Once these assumptions hold, the following questions are considered: How can the group benefit be measured compared to default simulation behavior, other groups or individual driving? Next to minimizing times, safety is a secondary concern of relevance in this thesis. Autonomous vehicles focusing on social group behavior will not be accepted unless the probability of collisions is extremely low and there are other benefits like less emissions or energy consumption. One approach to pursue towards benefits is the Gap Acceptance Model (GAP) solution, developed for this thesis with my colleague `Chu` [64].

According to `Klügl` ([207] p. 207), the dimensions of characterizing on the agent and system level are included:

- Number of agents: maximum 5275 agents (because this is the amount of vehicles that run in the base use case scenario of the Southern Part of Hanover)
- Level of Heterogeneity: homogeneous and four heterogeneous agent vehicle types
- Complexity of the agent architecture: rather simple focused on vehicle characteristics combined with Jason agent behavior
- Form of behavior: driving (accelerating, stopping, turning)
- Forms of internal memory: during simulation, in future learning parameters could be stored
- Types of organizational processes: small role-based leader-member setting in cooperative benevolent manner (no competition within or with other groups)
- Types of relations: individual and grouped vehicles being either leader or member
- Forms of Communication: indirect interaction (through movements and their correlating visual signals) and direct messages
- Locality of interactions: within the network, most likely within communication range
- Fixed or variable structures: fixed environment with fixed signal programs, variable vehicle distribution with a traffic density of category D - dense traffic.
- Feedback Levels: different levels of interaction through agent architecture and layers differentiating microscopic tactical planning, mesoscopic group coordination, and macroscopic strategic planning as shown in Figure 5.2.

3.7 Research Gap

For this thesis, the aim is a system to execute simulations of dynamic vehicle groups in urban traffic. Therefore, in the summary in table 3.2, systems or authors are compared by the characteristics of interest for fulfilling the scope of this thesis. All systems specify their behavior with structure, functions and/or deontic rules. Commonly, the focus is on traffic lights instead of vehicles for traffic control strategies. Since traffic is quite complex, the system needs to adapt to changes in a dynamic fashion instead of falling back on pre-calculated plan systems using preliminary defined rules based on historic data in a static way. The third focus is on decentralized architecture to deal with microscopic individual behavior where, commonly, macroscopic and global behavior is used for traffic management. Social behavior can diverge from individual behavior, which is also investigated. Communication enables participants to exchange information, and, lastly, systems can be commercial and closed or open and free to use and extend for new behavior.

The information in the table is abbreviated with the following notations: $-$ = *no*, + = *yes*, ? = no info given, *O* = Hybrid Architecture. Hybrid architectures include vehicles as well as control structures like traffic lights and usually represent an interaction between the low level of vehicles and the control instruments provided by the traffic management.

Table 3.2: System Summary with Relevant Characteristics.

System/ Author	Vehicle	Dynamic	Decentralized	Social	Communication	Open
DesJardins et al. [83]	+	+	+	+	?	?
Ortúzar & Willumsen [266]	+	+	+	+	?	?
deOliveira et al. [75]	O	+	+	+	-	+
Helbing et al. [161]	O	+	?	+	+	+
Mataric [245]	-	+	+	+	+	?
Yamashita & Kurumatani [385]	-	+	+	+	+	?
Wiering [378]	-	+	+	+	?	+
Bazzan [25]	-	?	+	+	+	+
Gipps [132]	+	-	+	+	-	+
Barceló et al. [21]	+	-	+	-	-	+
Junges & Bazzan [197]	-	+	+	-	+	+
Tumer et al. [351]	+	?	O	+	+	?
Wang [370]	-	+	+	+	?	?
Bazzan & Klügl [29]	O	+	?	+	-	+
Prothmann [278]	-	+	O	+	?	+
Roozemond [297] [296]	-	+	O	+	+	?
Jiang et al. [193]	+	+	+	-	-	?
Katwijk [360]	-	+	+	-	+	?
TRYS [73]	-	+	O	-	+	+
Nagel & Schreckenberg [259]	+	-	+	-	-	+
UTOPIA [246]	-	+	+	-	+	-
Oliveira et al. [74]	-	-	+	-	+	+
Schepperle & Böhm [307]	O	?	+	?	+	?
Steingröver [327]	+	?	-	?	?	+
de Palma et al. [77]	O	+	+	-	?	-

Balan & Luke [14]	O	-	O	+	+	?
Dresner [89, 91]	-	+	+	-	?	?
Bazzan et al. [27]	+	+	O	-	-	?
Rossetti et al. [298]	+	+	O	-	-	?
Mandiau et al. [240]	O	+	+	-	-	?
Bazzan et al. [26]	O	-	?	+	-	+
OPAC [120, 121]	-	+	+	-	?	-
deOliveira & Camponogara [76]	-	-	-	?	+	+
Jeon et al. [192]	+	?	?	-	+	?
Au et al. [9]	-	?	O	?	+	?
Vasirani & Ossowski [363]	O	?	O	-	?	y
Charypar et al. [61]	O	-	-	+	-	+
Gershenson [128]	-	+	O	+	-	+
Jennings [187]	-	?	-	+	?	?
RHODES [251]	-	+	O	-	?	?
Sueur et al. [331]	-	?	O	+	-	?
France & Ghorbani [116]	-	-	O	?	?	+
TUC [85]	-	+	-	?	?	-
CRONOS [44]	-	+	-	-	?	?
Irani & Leung [182, 183]	-	?	?	-	-	+
PRODYN [166]	-	+	O	-	?	-
Katwijk & Koningsbruggen [359]	+	-	-	-	-	?
Oberschelp et al. [264]	-	-	?	-	-	+
Kolodko & Vlacic [209]	-	-	?	-	-	?
SCATS [233]	-	-	-	-	?	-
TRANSYT [293]	-	-	-	-	?	-
SCOOT [179]	-	-	-	-	?	-

The first ten entries match in four out of six required characteristics and therefore influence the research for this thesis to a greater or lesser extent. The middle of the Table 3.2 represents systems and approaches which have three or then two matching criteria. The last twenty systems or authors are the base of knowledge, including many valuable approaches, for example for classical traffic control for the environment.

Please note that not all approaches which influenced this thesis are listed here, but are mentioned in other Chapters where direct referencing is needed. There are some within the model, the traffic simulation systems, and the case study of urban traffic groups.

Considerable amounts of environments, agent systems, interactions and organizations have been presented for their research effort dedicated to the exploration of how traffic simulation and decentralized autonomous vehicle groups can be reconciled. It represents a wide field of autonomous agents as a part of artificial intelligence influenced by control theory, organizational structures influenced by psychology and its group theories, and other approaches from eco-

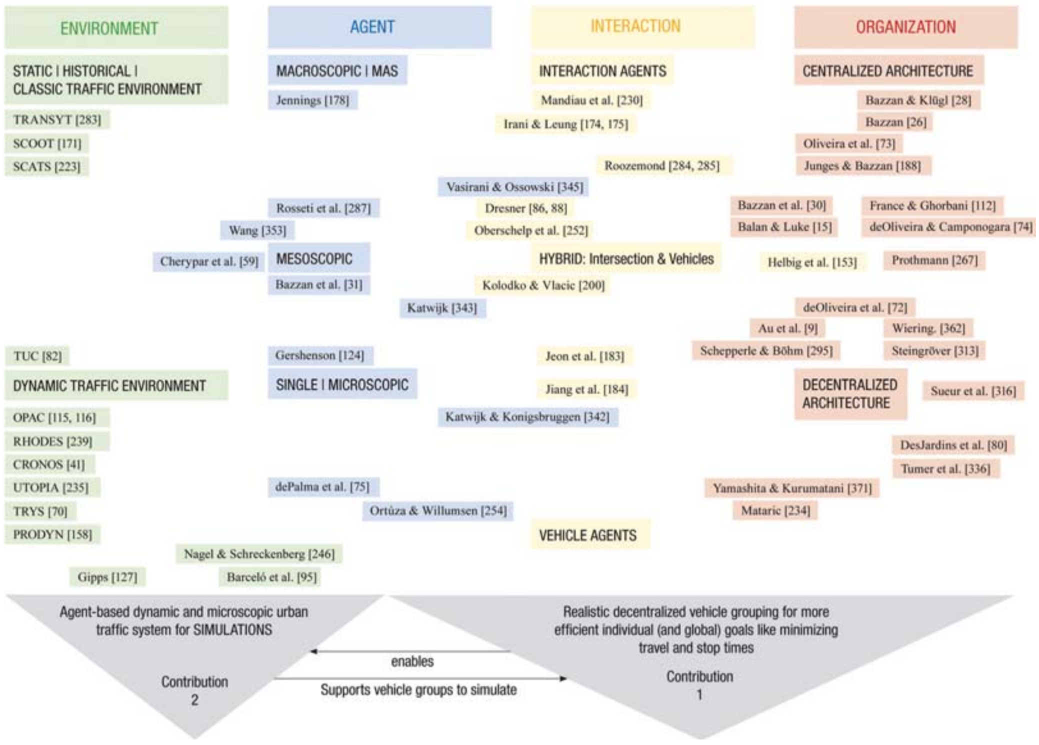


Figure 3.8: Decentralized Autonomous Vehicle Groups in Urban Traffic: An Area Map.

nomics, organizational theory, and sociology, which were discussed in chapter 2. On the one hand, there are virtually no approaches explicitly supporting the ability of microscopic traffic agents to interact, to organize themselves and co-operate with other agents within a simulation framework in a dynamic and open environment. On the other hand, grouping vehicles instead of the more centralized traffic lights as autonomous agents in urban traffic is not very common, and if so, vehicles are regarded as individual entities and formed into groups. This comes more from other domains like gaming for team play, biology with swarms, and robotics, which were out of the view of traffic agents.

Figure 3.8 illustrates a map of research for addressing autonomous vehicles groups in urban traffic and the simulation environments. The vertical ordering of the individual approaches is approximately by the biggest influence in those areas. On top are the more classic approaches whereas on the bottom are the more similar ones to my investigation goal of vehicle groups.

The outcome of this summary of the state of the art is that main objective to show the effects of autonomous vehicle groups in urban traffic. In order to do so, a suitable simulation platform needs to perform the simulation scenarios. This thesis provides two contributions (1) a model for autonomous vehicle groups in urban traffic to focus on decentralized participants and (2) a simulation environment linking interacting individual agents with the traffic environment called MATI.

3.8 Summary

Based on the above empiric observation that traffic control can privilege partial interests of traffic participants, a group-oriented, that is a mesoscopic theory, is developed, which engages the formation, description and group work process depending on group interests. The single individual by definition rarely has the possibilities to suggest, much less enforce its beliefs compared to the whole. Therefore, the significance of groups offers, in general, the possibility of influencing and modeling the whole traffic system as well as participants more efficiently. On the basis of a comprehensive literature survey, basic characteristics have been carved out, which serve as a reference to classic approaches in order to identify additional modules for the mesoscopic theory, which is open and holistic.

The design and analysis of simulated traffic systems and autonomous agents involves finding solutions to a large and diverse array of problems, concerning, besides the dynamic traffic management in urban areas, individual agent behaviors, interaction, and collective or even cooperative behavior. A wide variety of urban traffic scenarios and systems, and multi-agent approaches to these, have been studied in recent years. They have been presented in this chapter. These studies suggest models that support the design and the analysis for the urban network environments at the level of the traffic agents with or without interaction, and the organizational level of the multi-agent systems, especially cooperative behavior for traffic participants.

For reproducing real-world traffic simulation and controlling the traffic flow, agent-based technologies can be applied to traffic and transportation.

The primary goal of this thesis is to bring together novel work on autonomous vehicle groups sometimes already presented in papers. Traffic simulation, vehicular and infrastructure communication derive from diverse fields and are combined with computer science, more specifically artificial intelligence. Distributed and decentralized systems focus on modeling, implementation, and evaluation of autonomous vehicular agents and suitable simulation systems. Thus, the conventional approach to traffic simulation is extended by the power of autonomous vehicle group formation. Additionally, a simulation system (MATI) is established to evaluate vehicle grouping in traffic simulations.

If you realize that all things change, there is nothing you will try to hold on to.
Tao Te Ching (Lao-Tzu, Verse 74)

Chapter 4

Agent-based Traffic Management Systems

Chapter 4 introduces the evolution of agent and traffic simulators for the purpose of vehicle groups in urban networks.

Agent-based traffic management systems can use the autonomy, mobility, and adaptability of mobile agents to deal with dynamic traffic environments. This chapter shows the state of traffic control and management systems based on mobile multi-agent technology. Multi-agent systems (MAS) support fine-grained modeling of preferential and motivational states, knowledge and reasoning of individual traffic participants. Heterogeneous and changing structures, complex information processing, and decentralized decision making can be implemented considering multiple factors and dynamic information, enabling, for instance, grouping, learning and adaptive behavior. Also, MAS inherently supports a distinction between the micro, meso and macro perspectives in simulation and thus can provide a unified perspective on multi-level simulation tasks. The problem, however, is how to bring together agent-based modeling with traffic simulation. An avenue to solve this problem is to create an agent-based simulation platform from scratch like in Section 4.3.1. However, doing this for the complex traffic domain is a cumbersome task, because existing traffic simulators come with a wealth of domain models and libraries (e.g., models of pedestrian behavior, drivers, car following or fluid dynamics), and rich visualization and analysis tools like the *commercial* traffic simulators AIMSUN or *open-source* SUMO, as listed in chapter 2.2.5.

The *centralized* systems have their strengths in reliability, robustness, and maturity e.g., a study of traffic networks TRANSYT [293], traffic signal optimization SCOOT [179], and a more adaptive traffic-responsive approach SCATS [233].

This research focuses on decentralized and dynamic approaches and in detail on the microscopic level of individual vehicles such as in own work of `Fiosins et` `al.` [106] 'Agent-Based Integrated Decision Making for Autonomous Vehicles in Urban Traffic' and Görmer et al. [137] 'Decision Support for Dynamic Traffic Management Using Vehicular Communication'.

Intelligent transportation can provide services such as decision support for individual traffic participants and a standard development environment for traffic management strategies, but served as a testbed for the research that went into this thesis.

This chapter continues with *requirements* for and *comparison* of agent-oriented simulation for cooperative traffic. The section Own Approaches introduces several Agent-based Traffic Management Systems: (1) Streetworld as a Java-based Agent Traffic System from scratch, (2) AIMSUN extended via API and SDK with Dynamic Traffic Management, communication, data and agent technologies of learning and grouping, (3) ATSim, a combination of the commercial traffic simulator AIMSUN and the agent interpreter JADE and, finally, (4) MATI, a framework for coupling different simulations with agent platforms. Both traffic simulators and multi-agent systems can be exchanged and employed with analyzing tools.

4.1 Requirements

In developing a simulation environment - including a traffic simulator with connected agents - to pursue the research goals of this thesis, the aim was to satisfy the following criteria (compare idea thesis Vasirani [362] p. 102 but adapted to own purpose and goal):

- **Agent Accuracy:** Traffic is an emergent phenomenon, the result of the individual decisions of drivers, traffic controllers and all the actors that are part of the system. Agent-based modeling helps build simulated models with detailed, rich behaviors for individual entities. For this thesis' purpose of vehicle groups, each agent of the system must have a certain degree of autonomy (especially the driver agent) and it must be possible to program their behavior and decision making.
- **Integration with traffic control:** Borrowing the terminology of the theory of control, the simulator must combine the intelligent vehicle agents with the simulated traffic control system (i.e., streets, intersections, traffic lights), to enable experimentation with different agent and environment control mechanisms.
- **Small-scale simulation:** In this research the focus is on individual and small group coordination mechanisms. This simulator must be able to simulate up to thousands of entities with a high degree of detail and accuracy and a good computational performance.
- **Realistic dynamics:** The aim is to simulate up to thousands of agent vehicles traveling through different pre-defined artificial and realistic road network scenarios at a certain density in rush hour (Level of Service C-D). This ensures the possible application to academic comparisons of models and realistic traffic in cities.

The simulator must employ validated traffic models in order to simulate realistic dynamics of the vehicles. The measurements of default simulation are going to be compared to the modified autonomous vehicles driving in groups.

These four above-mentioned desiderata are the functional requirements. The following technical or non-functional requirements need a closer look. The characteristics, properties and facts according to which the criteria are collected need to be defined and described. This is done with the Factors-Criteria-Metrics-method (FCM) and ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models [184]:

- **Functionality:** with the criteria of suitability, correctness, interoperability, constancy of requirements, accessibility
- **Reliability:** with the criteria of maturity, fault tolerance, recoverability
- **Usability:** with the criteria of comprehensibility, learning ability and usability
- **Efficiency:** with the criteria of time and resource behavior
- **Maintainability:** with the criteria of being analyzable, changeable, stable and tested
- **Portability:** with the criteria of being customizable and installable, compliance with norms and conventions of portability, and replaceability.

4.2 Comparison: Agent-oriented Simulation for Cooperative Traffic

Next to agent-oriented simulation platforms there is also pure transport simulation software like AIMSUN or SUMO. Those vehicles - in the form of agents - are representable, but the possibility of modification and programming of the vehicles is not provided. This is problematic for autonomic vehicles, because, in the form of agents, extra agent software is needed, and, because, for possible approaches of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication, an extra software add-on like OMNeT++ (`omnetpp.org`) is required.

Approaches like ATSim, described in Section 4.3.4, and MATI, described in Section 4.3.5, combine agent-oriented simulation with transport simulation and must be connected with an external interface. The programming effort and domain knowledge is relatively high compared to only agent-oriented simulation for cooperative traffic. Other problems of the combination of traffic and agent software are performance quality and scalability fall-off. This is due to big data exchange between the two simulations which need to be received, processed and sent, so the next simulation step has a longer transfer duration with applications of many vehicle agents.

An extensive evaluation was conducted for exclusively agent-oriented simulations which offer a broader application field and functionalities, but are independent from other software. However, the traffic specific requirements which are offered by a state-of-the-art transport simulation system like AIMSUN or SUMO are fundamental for the generation of a traffic scenario within an agent-oriented simulation platform. Some of those traffic specific requirements are adopted for the catalog of requirements (described in the following Section B) for agent-oriented simulation for cooperative traffic. In order to understand the processes of traffic control and dynamics further work is necessary. An important option is the further development and modification of existing software.

Eight agent-oriented simulation platforms: (1) AnyLogic, (2) JADE, (3) Jason, (4) MASON, (5) MATSim, (6) NetLogo, (7) Repast Simphony, and (8) SeSam platforms were evaluated and are attached in Appendix B.

For scenarios where communication and the modeling of complex agent behavior are preferred, Jason and JADE are good options.

For grouping vehicles especially individualization and interaction are especially needed and the environment and algorithms could also be done with tools from the traffic simulation for more mature results. Therefore, the preferences for this thesis are Jason or JADE as agent interpreter. Although, for very realistic environments in a simulation an integrated algorithm for individual and joint behavior including communication features is needed.

4.3 Own Approaches

Throughout the thesis several approaches were pursued from scratch up to combining existing platforms.

4.3.1 Streetworld: Agent-based Traffic Simulation

Simulations are useful for testing and developing ideas and strategies without much effort. Multi-agent systems are a new field in computer science. In `Wooldridge` [382] (p.5) simulations are based on the acting objects (the agents), which are equipped with a goal or plan and interact in an environment:

> A multi-agent system is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure. In the most general case, the agents in a multi-agent system will be representing or acting on behalf of users or owners with very different goals and motivations. In order to successfully interact, these agents will thus require the ability to cooperate, coordinate, and negotiate with each other, in much the same way that we cooperate, coordinate, and negotiate with other people in our everyday lives.

In this Streetworld environment agents shall cooperate, coordinate and negotiate with each other or with the environment and try to reach their goals.

Streetworld is an agent-based simulation platform for testing and developing strategies for cooperative traffic management, for example, how vehicles group themselves in order to coordinate and negotiate their path through a green wave of traffic lights. Also other grouping behaviors are tested for their influence on the traffic flow of each single vehicle as well as for groups and for the net distribution as a whole. The decentralized perspective is the focus on the single vehicle and resulting groups. It is assumed that through grouping mechanisms the micro and macro behavior can be improved. Different group algorithms which will be implemented should show this in Streetworld.

Figure 4.1: Streetworld is a traffic simulation using Jason `http://streetworld.sourceforge.net/`

In the following the structure, function and aims of Streetworld are described. In order to understand the Streetworld operating mode, previous knowledge in the agent interpreter Jason and the corresponding environment CArtAgO is advantageous. For those frameworks there are useful tutorials and introductions in existence which is why they are not further described here.

Streetworld is based on Jason and is a Multi-agent System. Its simple three-lane scenario is shown in Figure 4.1. Through simulations the behavior of agents can be investigated, which are implemented with AgentSpeak. Jason provides an environment for those simulations where the agents can interact. But this environment is not flexible enough for Streetworld and cannot fulfill the needs of a traffic simulation. For that reason, Streetworld uses its own environment.

This image 4.2 shows the main features of the Streetworld environment. The core of the environment is the ModusArtifact. This class contains all scenario-specific information and acts as an interface for the agents with the 'outer world' as well as for the agents with each other. The communication medium is the CommPort so that the agents can interact with the ModusArtifact. ModusArtifact and CommPort inherit from Artifact, a class of the CArtAgO-Framework. CArtAgO is an extension of Jason which supports the environment perception and simplifies the interaction between the agent logic of AgentSpeak and Java objects. For example, it is possible to call Java objects with agents, so called artifacts, as well as define access rights for other agents and call their methods. The CommPort contains agent specific information and is only accessible and controllable by this agent.

Installation

In this section the presetting is described with which to program in and with Streetworld.

Streetworld is based on Java. OracleJDK or OpenJDK are mandatory. SVN is used as a revision control system and is necessary for the download

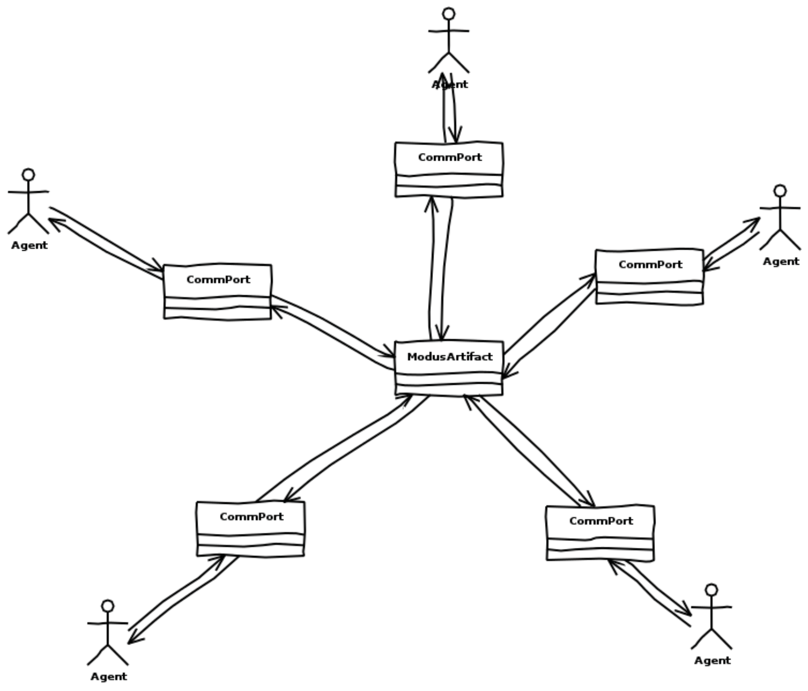


Figure 4.2: Streetworld Environment.

of the source code. Streetworld was developed with the Build-Management-Tool Maven Version 2.2.1 and therefore it is a prerequisite. The use of Eclipse with the plug-ins Jasonide and m2e, an integrated development environment is optional. For the graphics BadLogicGames 0.9.4 are used for the graphics and are necessary.

After fulfilling all the software requirements, Streetworld can be downloaded:

```
svn checkout svn://svn.code.sf.net/p/streetworld/code/trunk/streetworld
```

Subsequently the jar-file is retrieved with corresponding directories for the sections Jason, Java Home and Ant libs:

```
java -jar [Streetworld_Dir]/lib/net/sourceforge/jason/jason/3.6a/jason-3.6a.jar
```

The other sections are not required. Jason saves this general information in the user directory under

```
 .jason/user.properties
```

Now the installation can be tested. In the Streetworld directory the code needs to be compiled and then started:

```
mvn compile
mvn exec:exec
```

The latest executable version of Streetworld can be downloaded via Sourceforge as an archive.

Simulation Scenario

The goal of the Streetworld project is to develop a skeletal structure for traffic scenarios. With this base especially group behavior of vehicles should be simulated. Streetworld uses the CArtAgO extension of Jason for creating the traffic environment.

First achievements are the implementation of a Color-Sort-Algorithm (CSA) and rudiments of the Manhattan Distance Function (MDF). In the initial state of the CSA different colored vehicles (stands for different properties like technical abilities, goal, route etc. - similar to Figure in 4.1) are distributed among the available lanes. When starting the simulation the vehicles start driving and try - as efficiently as possible - to change lanes in such a way that the result is that only vehicles of the same color are driving in each lane. The MDF can be useful in a scenario in order to find similarities of vehicles (such as same desired speed, acceleration or deceleration as technical abilities of the vehicle) in order to group those vehicles according to their similar preferences. This causes vehicles with of the same vehicle type such as slow, medium or fast preferences to group. The more similar the group members are, the more stable the group is. Bigger differences are needed for different groups to be stable. For example, vehicle types are in groups of normal cars, truck/bus, fast cars and small slow cars.

A simulation scenario, called SimScenario, is a predefined initial situation - here a conflict - of agents which is loaded and executed by Streetworld. For creating a SimScenario the file needs to be generated:

```
[Streetworld-RootDirectory]/SavSimScenario/[SimScenario-name].simszen
```

This file is interpreted by Streetworld as a property file. Here is an example:

```
# Streetworld Scenario

name             = DriveTest1
name_s           = DT1
modus            = Modus_CS
modus_package    = org.streetworld.modus.CS
window_gdx_size  = 960 70
screen_gdx_size  = 1920 140

#                    x  y dirx diry vTyp agTyp color
robo_list        = 100  1    1    0    2     3     0 \
                   100  2    1    0    2     3     0 \

other_agent_list =  -1 -1   -1   -1   -1     4     1

#                  typ   x y
map              =   0   0 0                         \
                     0 960 0
```

Explanation of terms used:

- *name:* identification for the SimScenario.
- *name_s:* Stands for short name. This is used as part of agent-identifiers.
- *modus:* class name of the modus used.
- *modus_package:* package description of the modus used.
- *window_gdx_size:* size of the simulation display window. X- and y-values can be separated with space or tab.
- *screen:gdx:size:* size of the graphical output within the display window. x- and y-value can be separated with space or tab.
- *robo_list:* All vehicle agents are defined. Agents are defined in line-by-line with the following properties:
  - *x:* x position of the agent
  - *y:* y position of the agent
  - *xdir:* x ratio of the agent alignment
  - *ydir:* y ratio of the agent alignment
  - *vTyp:* stands for vehicle type. The types can be chosen by the corresponding integer: 0 = car; 1 = van; 2 = truck
  - *agTyp:* stands for agent-type. [the Java codes need to be adjusted for it]
  - *color:* stands for color or cluster group, that is a group with the same or similar properties [the Java codes need to be adjusted for it]
- *other_agent_list:* those agents which are not vehicle agents or should not be counted as such are defined in this list. The syntax of the definition is similar to robo_list except that non-existant or not needed properties are flaged as -1 (not defined).
- *map:* the environment map is defined by type and x and y coordinates.

Figure 4.3 shows a startup diagram of which processes are running until the scenario is loaded completely. The agent environmentCreator is specified in streetworld.mas2j and therefore is started by Jason automatically. The StreetworldGUI inherits from MASConsoleGUI and is defined in logging.properties as a handler, thus, also loaded automatically. All other scenario specific classes are started as shown in Figure 4.4:

The core of the view system in Figure 4.4 forms the streetMap which is an array list where the ModusArtifact is defined and initialized. It contains objects of the class ValueBox_Street. Those value boxes are defined in the CommPorts of the agents and connected with the streetmap. They contain all the important attributes for the view system such as the identification, the position, the length and the flashing state of an agent. The flashing state - administrated by the CommPort itself- shows whether an agent is indicating and if so on which side. The identification and length are fixed values. The current position is administrated in the sprite object (see graphic output in Figure 4.4). With the

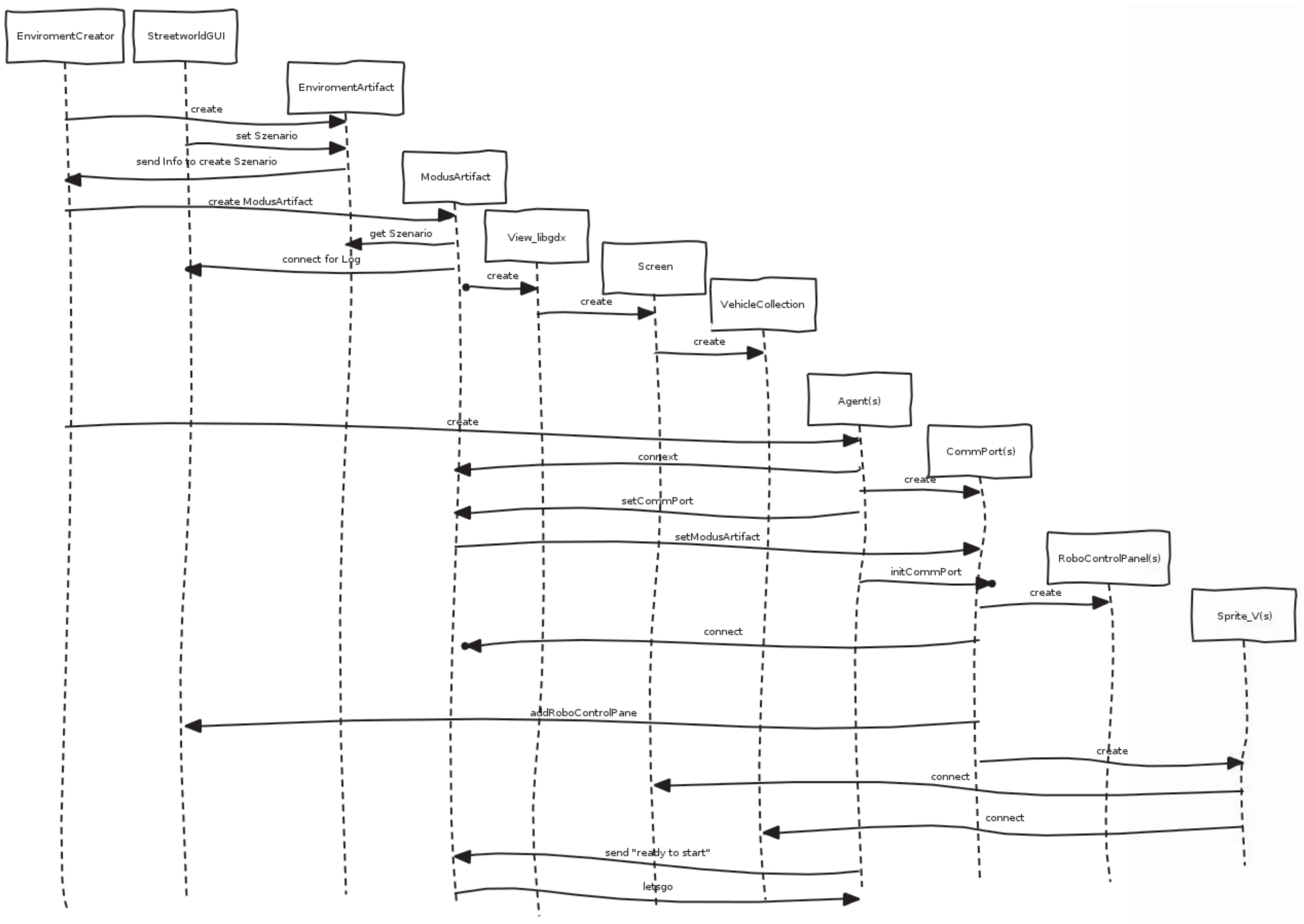


Figure 4.3: Startup Diagram of Streetworld.

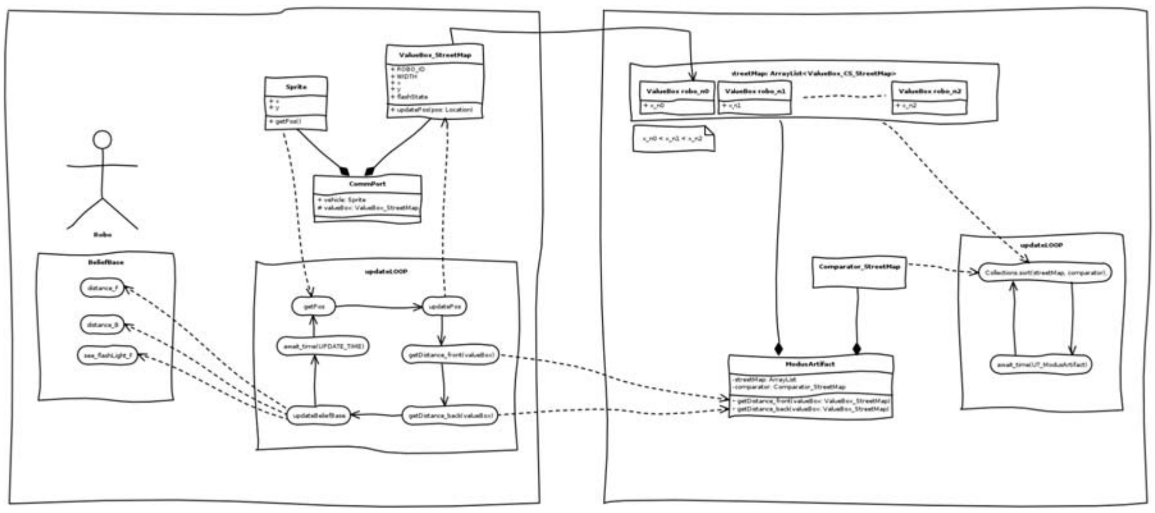


Figure 4.4: The View System of Streetworld.

Table 4.1: Streetworld Summary Information.

Field	Value
Name	Streetworld
Description	Traffic simulation using Jason
Website	http://streetworld.sf.net
GroupId	org.plathe.streetworld
ArtifactId	streetworld
Version	0.0.2-SNAPSHOT
Typ	jar
JDK	Rev 1.5

method update() all attributes of the valueBox are refreshed. This INTERNAL_OPERATION (see CArtAgO/Internal Operation) works as a thread which contains a loop with the stopping time. The ModusArtifact has a similar update method which sorts continuously by ascending x value of the updated value boxes of the agents in the streetMap with the help of Comparator_StreetMap. In the update() method of the CommPorts of the agent the methods of the class ModusArtifact getDistance_front(ValueBox_CS_StreetMap) and getDistance_back(ValueBox_CS_StreetMap) are called. Those methods give the distance to the vehicles in front of and behind the agent in all three lanes. Behind the agent meaning behind the front of the agent, not necessarily behind the complete agent. If the distance to the next vehicle is bigger than the VIEW_RANGE, a value which is defined in the ModusArtifact, the distance is returned as -1 unknown. The method getDistance_front(ValueBox_CS_StreetMap valueBox) also returns the flashing status of the vehicle after the distance has been calculated. The flashing state of the vehicle behind is not important for this scenario. Finally with the method getObsProperty(String).updateValues(Object) in the update loop of the CommPort, the BeliefBase of the agent is renewed.

Status Quo

Streetworld is an open source project in Sourceforge. Due to the missing traffic dynamics available and the goal of this thesis, Streetworld was stopped in 2013. The reason was that all need to be implemented by hand with mathematical models of traffic dynamics which is not the main focus nor expertise, but rather the individual vehicle agent and group behavior. The focus is vehicle groups with agents not traffic dynamics.

Substantial tasks for Streetworld which were still to be implemented:

- Manhattan-Distance-Function rudimentarily done
- group forming algorithms and evaluations:
  - Color Sort: from front and back is implemented
  - Contract Net Protocol
  - Joint Goals

  - Shared Plans
- implement Nagel-Schreckenberg

Technical tasks for Streetworld partly done, but still to be implemented:

- write and read scenarios in normal text file - done!
- use of property files
- find a solution for the use of relative file paths: at present the script starting the program jumps into the to root directory of the application. Therefore it is possible to access property files etc. A better solution with .class.getClassLoader() should be found.
- introduce ChangeLogs

Another similar project called MovSim (http://www.movsim.org) by Martin Treiber and Arne Kestings was found, which has more advanced scenarios where the Nagel Schreckenberg celluar automata model, among others, is implemented. The only problem is that it is also hardly active 2013 anymore - 'just as a hobby in progress'.

Streetworld Discussion

Of the four non-functional requirements described in requirements 4.1, only one and a half are met:

- **Agent Accuracy:** This thesis' purpose of vehicle groups is only partly fulfilled by two simple color sort algorithms which were implemented in Streetworld. The agent aspects of the system are inherited by Jason and CArtAgO and therefore have the degree of autonomy needed, especially the vehicle agents - the robos. Their behavior and decision making is programmed in the AgentSpeak language.
- **Integration with traffic control:** intelligent vehicle agents can be created but the simulated traffic control system with streets, intersections and traffic lights is very hard to create. A gaming engine was used for smooth visualization. Experimentation with different agent and scenario environments were only done as a prototype. The project was stopped because the traffic control requirement was not met.
- **Small-scale simulation:** Streetworld has the ability to simulate up to 100 agents which could could partially meet the requirement partially to form small vehicle groups in a network. Just one street with three lanes was implemented as an scenario environment.
- **Realistic dynamics:** One pre-defined artificial road network with three lanes was implemented, but with no realistic density. More scenarios would need to be implemented. But the Streetworld simulator does not employ validated traffic models like Nagel-Schreckenberg. This would still need to be done. Also, models of vehicle agents driving in groups are only implemented with two simple color sort algorithms, which does not provide a realistic simulation.

The following non-functional requirements are met only partly:

- **Functionality:** Streetworld is only partly suitable and interoperable, because vehicle groups are only implemented with two color sort algorithms and not more groups like the Manhattan Distance Function for checking similarities of vehicle characteristics. Streetworld works correctly with its limited scenario, but relative paths can be improved. The functional requirements are not met (just 1.5 of 4), but Streetworld is accessible as an open source project via sourceforge: `http://streetworld.sourceforge.net`
- **Reliability:** Streetworld is just a very basic prototype and far from mature. Fault tolerance is not looked at but Streetworld can be recovered.
- **Practicability:** Streetworld is simple to understand and, with prior knowledge of Jason and CArtAgO, easy to learn. The use is intuitive.
- **Efficiency:** The criteria of time and resource behavior has not been investigated.
- **Maintenance:** The criteria of being analyzable, changeable, stable and tested, has not been investigated.
- **Portability:** Since Streetworld is Java-based, it is customizable and an installation is provided. Compliance with norms and conventions of portability and replaceability is not investigated.

All in all Streetworld was an attempt to meet the goals of this thesis with a small simulation were everything is known and can be manipulated. Jason seemed to meet the agent requirements, but especially goal of the traffic control is not met and is too complicated to meet when not coming from this domain. Streetworld is a small prototype to ”playing” with agents from scratch, but it has not been active since 2013. Streetworld needs more traffic dynamics and the potential for analyzing vehicle groups is too low. More specialist knowledge of environment (traffic flow and dynamics) and algorithms (Nagel-Schreckenberg, car following models) is needed.

Streetworld anticipates that requirements to agent traffic simulators are underestimated. There is just one existing street and extending it would entail a lot of effort. Only rudimentary driving behavior is respected and existing algorithms need to be implemented. In discussion with `Arne Kesting` [348], the driving behavior was classified as being unrealistic. Also the graphic engine is not used optimally which results in instability and high processor load.

4.3.2 AIMSUN Extended

This section is based on the Simultech [137] and ITSC [108] conference paper and work within the project ’Planning and Decision Making in Networks of Autonomous Actors in Traffic (PLANETS)’ funded by the Lower Saxony University of Technology (NTH).

Within the PLANETS project and corresponding research, the base was the commercial microscopic traffic simulator AIMSUN (due to previous experience).

AIMSUN models the realistic dynamic traffic environment of the southern part of Hanover, Germany, combined with other domain knowledge of data modeling, vehicular and infrastructure communication and multi-agent systems.

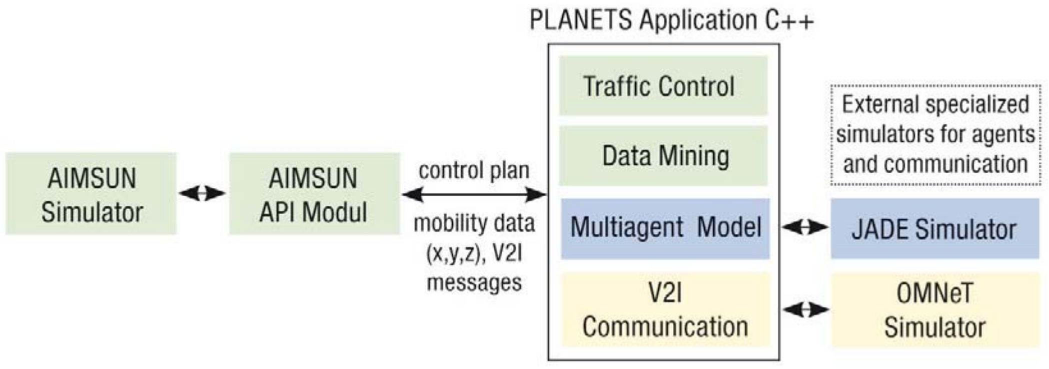


Figure 4.5: AIMSUN extended: AIMSUN traffic simulation with extension via the API module in traffic control, routing, multi-agent model and vehicular communication.

AIMSUN extended is an integrated simulation approach featuring centralized and decentralized traffic management in urban areas with the domain knowledge of dynamic traffic management, data modeling, communication and multi-agent systems. The aim is to improve traffic flows by dynamic traffic management which is supported by vehicular communication interlinking centralized and decentralized decision making.

The centralized decision component is to influence local traffic states with historic and future prognosis of traffic state estimation and the optimization of traffic lights - domain knowledge is needed for traffic management, data modeling and communication. On the other hand, decentralized decision making models individual traffic participants' behavior - domain knowledge of traffic management information input, traffic data, communication and multi-agent systems is required.

Another important property of the vehicles is their capability of autonomous communication and group formation, which makes it possible to coordinate their speed and make corresponding recommendations to drivers. Communication between vehicles is based on vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) protocols, which connect Traffic Management Center (TMC) and vehicles. The system is simulated using real-world data in AIMSUN software, which is extended by the decision logic of TMC and vehicles. Special attention is payed to the routing and grouping decisions of the vehicles, supported by corresponding communication.

Total interaction usually brings too much variety to simulation and is not considered in most standard traffic simulations like AIMSUN or SUMO. But new technologies in Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication are under development and usually have specialized tools like OMNet++. From the agent perspective the communication standard is FIPA ACL. This is supported by the middleware JADE or Jason including the extension A combination of Jason, CArtAgO and MOISE (JaCaMo).

In AIMSUN the simulator's API module is used and extended by its own application layer model and 'Radio Propagation' communication model specifically for urban environments.

The first is the application layer for managing and implementing functionalities between the following applications, seen in Figure 4.7 Learning App, Routing App and Grouping App, especially for handling and distributing messages between them.

The second is described in the following quote from `Görmer et al.` [137] p. 330:

> Radio Propagation Model: Typically, wireless network simulators assume a generic propagation model, such as the Free Space or Two-Ray Ground reflection model. While the former is a completely idealized model, the latter considers the effect of earth surface reflection and can be more accurate. However, neither of them considers the effects of the surrounding topology on radio propagation, which is especially important in urban environments. Therefore, models which capture predictable shadowing effects are appropriate for modeling urban vehicular communication, where the effects of buildings should be taken into account.
>
> In (Sommer et al.,2010), the authors present a computationally inexpensive simulation model for radio shadowing in urban environments based on real world measurements, which comprises an estimation of the effects that buildings have on the radio communication between vehicles. We combine their general model with a Nagakami propagation model for determining the received signal power level. Based on the calculation of received signal power at each receiver, the arrived packets are determined to be successfully received or lost.

`Görmer et al.` [137] state that, assuming vehicles are equipped with communication devices, those will broadcast Cooperative Awareness Messages (CAMs) using a repetition interval of 500 ms periodically according to ITS-G5 specifications [180]. CAMs include status information like station ID, position and speed, as well as the driving direction. All received CAMs are analyzed and dispatched locally for continuous use. This is useful for detecting neighboring vehicles located within a single hop distance and for the estimation of current traffic conditions. With those CAMs, no extra traffic detectors like video cameras or inductive loops, as shown in `Tchouankem` [338] are needed, because the world can be perceived only through communication. Figuratively, the communication can be seen as a postal service and in the AIMSUN project PLANETS it is used as the perception of the environment. Communication modeling is limited to AIMSUN's granularity of simulation time with a minimum time step length of 100 ms, but in IEEE 802.11 frame durations and MAC timings are in the range of μs.

To illustrate and validate the AIMSUN extended approach, a typical use case of a city road network is simulated. The microscopic traffic simulator AIMSUN was significantly extended using the AIMSUN API for simulations.

Installation

The commercial traffic simulator AIMSUN[1] needs to be installed. The project PLANETS started with AIMSUN Version 6.0.6 Expert Edition (upgraded later to Version 7.1) which is downloaded after buying as a zip file and needs to be unpacked. The following order of steps needed to be done for installation:

1. Installation of Aimsun_6_0_6_20091028_halc.exe
2. Installation of the Patch Aimsun_6_patch_20091106_lice.exe
3. Copy both licenses license_*.dat into the program folder TSS-Transport Simulation Systems \Aimsun 6 \licenses
4. if a server license is used: rename the server license in license.dat
5. open the browser and open the local host with port 1947 (make sure firewall opens in both directions)
6. choose *configuration* and go to the rider *Access to Remote License Managers*, then check 'Allow Access to Remote Licenses' and 'Broadcast Search for Remote License'. Put the IP of your License Server in and submit. (The folder also contains extra packages SDK and MicroSDK for developing purposes.)

AIMSUN is a very powerful traffic simulator. Besides microscopic simulations it also handles mesoscopic and hybrid simulations in different editions: Small, Standard, Professional, Advanced and Expert. Adaptive control interfaces are used with the microscopic simulator: SCATS, SCATS-RMS, SCOOT, Fast SCOOT UTC, UTOPIA, VS-Plus, as customized developments, and ETRA, INDRA, SICE, Telvent, Telent and ZGZ Prio with the mesoscopic simulator SCATS and UTOPIA 3.

The PLANETS project experienced problems with extending the traffic infrastructure with agent or communication technologies as shown in Figure 4.5. This was also due to the TSS support which was long to communicate with the developers, finally leading to a hard-wired solution using the AIMSUN SDK and API programming in C++.

4.3.3 Simulation Scenario

The AIMSUN simulator system achieves its goal of global settings influencing local decisions on the basis of information exchange: traffic participants provide information (location, speed etc.) to the TMC via floating car data (FCD) messages, allowing the TMC to produce a model of traffic states, estimate level of service (LoS, A-F) [13] and make decisions about the appropriate traffic control strategies. TMC sends its preferences of street use in the form of weights

[1] *AIMSUN* available at http://www.aimsun.com - 30 days trial

to the traffic participants at the same time. The traffic participants combine the received weights with their local ones, which represent their preferences of street usage. The resulting individual weights are used by traffic participants for decision making (e.g., routing). The traffic participants can form groups that coordinate their decisions. Vehicular communication enables linking of centralized control strategies with decisions of traffic participants. V2I communication supports communication between traffic participants and TMC in both directions. V2V communication supports a direct connection between the traffic participants and is used mostly for group formation and group coordination.

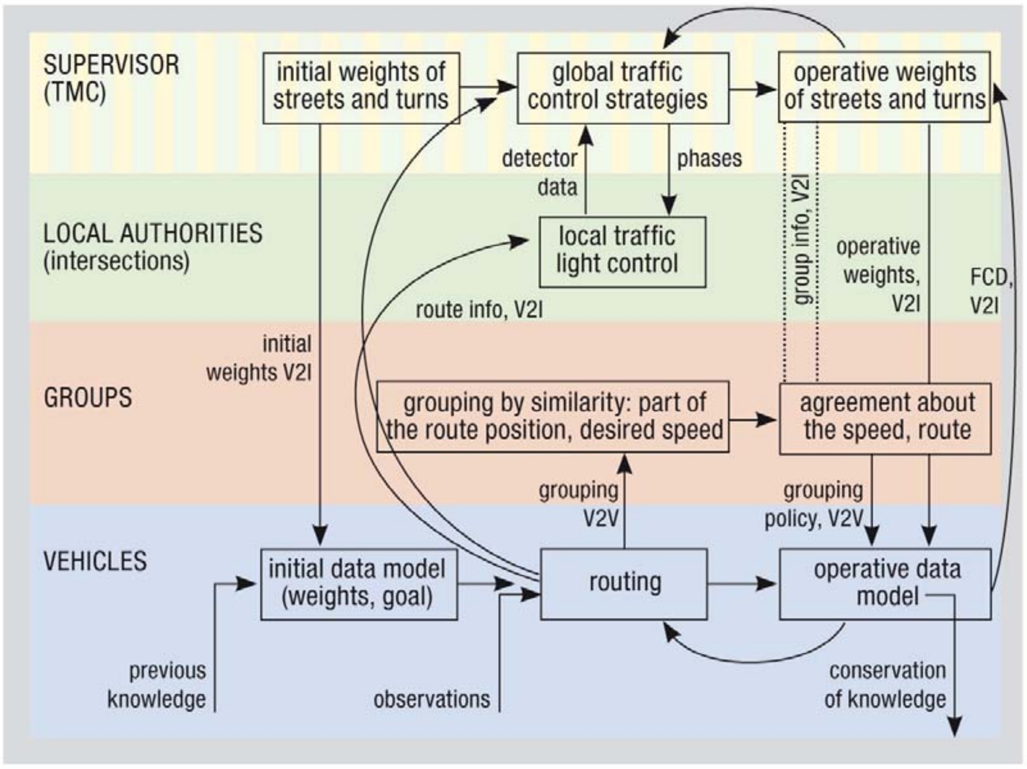

SUPERVISOR (TMC)
initial weights of streets and turns
global traffic control strategies
operative weights of streets and turns
detector data
phases
LOCAL AUTHORITIES (intersections)
local traffic light control
route info, V2I
group info, V2I
operative weights, V2I
FCD, V2I
initial weights V2I
GROUPS
grouping by similarity: part of the route position, desired speed
agreement about the speed, route
grouping V2V
grouping policy, V2V
VEHICLES
initial data model (weights, goal)
routing
operative data model
previous knowledge
observations
conservation of knowledge

Figure 4.6: AIMSUN extended: Interplay of a traffic scenario.

Figure 4.6 provides a comprehensive microscopic traffic simulation framework. Innovative functionalities regarding dynamic traffic management, decentralized decision making, as well as realistic communication modeling and simulation are featured. The interplay of a traffic scenario takes place on different levels: the *supervisor*, namely the Traffic Management Center, *local authorities*, here the intersections, *groups*, and *vehicles*. Before the initialization of a scenario each vehicle gets the initial weights of streets, turns from the supervisor and stores it as its initial data model together with previous knowledge and its goal. During the traffic scenario, the supervisor decides on global control strategies based on detector data of the local traffic light control and returns new signal phases and changes the operative weights of street and turns accordingly. Later the supervisor can develop new global control strategies including information of Floating Car Data (FCD) and Vehicle-to-Infrastructure (V2I) data. The vehicles decide on their route based on observation (what they have learned). With Vehicle-to-Vehicle (V2V) communication the vehicles decide to form groups based on similarities: part of route, position and desired speed. They join the goup and agree on the joint speed and route and give that information back to the individual vehicle along with grouping information, the policy and vehicular communication. They can also change their operative model where they can learn for their next routing decision. Another influencing factor of the vehicles' operative data models are the operative weights and V2I from the TCC. The vehicles driving information gives feedback to the supervisor, along with the FCD and V2I, and the knowledge is stored for future simulations. The dotted arrows are the future plans where routing information influences local traffic light control and also the global control strategies with the route info and V2I information. Another future feedback can come from the vehicle groups with their group information and V2I information, which could then change the operative weights and, for instance, give guarantees to vehicle groups for a green phase.

The AIMSUN Extended scenario as described above is demonstrated in Figure 4.6 and as a data flow in Figure 4.8. The extension is implemented on top of the AIMSUN traffic simulator, a commercial software for modeling complex traffic networks. The core of the simulator includes four API modules, which implement TMC (traffic control), traffic participants (represented as a MAS), routing services used by traffic management and participants, as well as a communication module, which implements realistic V2X communication.

The architecture of the simulator is presented in Figure 4.5. Vehicles are extended by agents' functionalities of reasoning about the environment and taking actions in it, i.e., vehicles are autonomous in making their routing decisions. They receive information from the environment in the form of messages from the TMC. Information available to the agent is stored in the form of local weights, representing their preferences and the information about the environment. The state information is then translated into routing decisions using the information about destination and corresponding routing services. The routing decisions are translated into corresponding actions (lane changes, turns etc.).

The architecture of a vehicle agent is shown in Figure 4.7.

A (simulated) vehicle is equipped with the following modules (Apps):

- Learning App - adoption of the weights provided by the TMC and their combination with the individual weights
- Routing App - access to the routing services
- Grouping App - group formation and corresponding decisions
- CommBox - communication module (V2I and V2V)
- setRoute App - translation of a route to corresponding actions (turns)

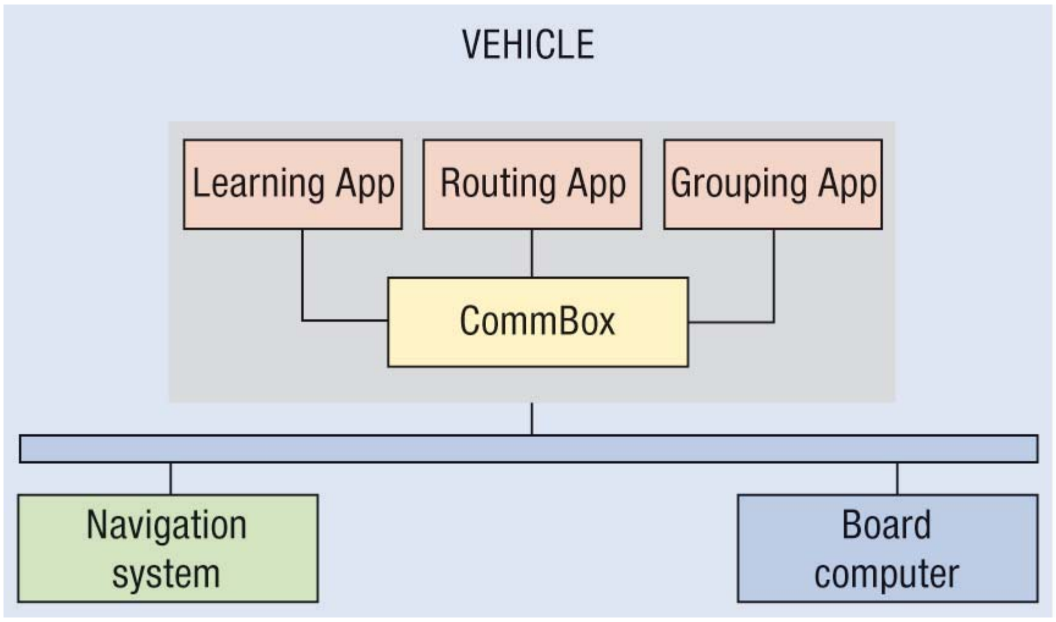


Figure 4.7: Application system of a vehicle agent for AIMSUN Extended.

In a simulation scenario, real-world traffic flow data is used. The traffic network is a model of the southern part of Hanover (Germany) with two parallel and five perpendicular streets (see Figure 2.15).

Our goal was to demonstrate the effect of the proposed combination of centralized control with decentralized decisions made by the traffic participants.

The Hanover scenario in Figure 2.15 demonstrates efficient routing in urban road networks by centralized and decentralized decision support. Communication ensures the distribution of information required for centralized and decentralized decisions. The focus is on the implementation of simulation functionality which integrates traffic simulation, realistic communication and agent decision making.

An area of tension arises from the combination of centralized control by dynamic traffic management and decentralized decisions by autonomous vehicles in the form of agents. This is investigated with regard to the impact of centralized and decentralized decision support in the morning rush hour from 7:30 to 8:30 a.m. Here, at least one junction is regularly overloaded (compare with the upper right corner in Figure 2.15), leading to traffic jams and significant extension of individual travel times. The minimization of travel times benefits the traffic management and the individual vehicles globally.

The aim is to automatically identify junctions suffering from bad traffic quality and dynamically adjust their traffic signal programs. Then, the autonomous vehicles receive via communication a centrally predefined rerouting strategy. The individual vehicles evaluate whether this new information applies to their route and can redefine their route appropriately depending on their perception of the traffic state. Thus, congestion at a crowded intersection can be avoided by spacious rerouting.

Technologically, it is assumed that traffic signals are centrally controlled by fixed signal programs corresponding to the current traffic state. Vehicles are equipped with a navigation and communication system, and communication infrastructure such as road side units is scattered around the main streets. V2I communication ensures communication between individual vehicles and traffic management, V2V allows the communication between individual vehicles.

An abstraction of the road network in the southern part of Hanover is implemented in the traffic simulation software AIMSUN. AIMSUN allows for the detailed modeling of city road traffic in terms of road infrastructure, behavior of the vehicles, traffic light control etc., embedded in a microscopic traffic simulation. Traffic flows and traffic signal programs are parametrized according to data from empirical traffic data collection as well as control programs in operation. Whereas AIMSUN features the precise modeling of single vehicles, traffic lights and urban road infrastructure, the simulation of vehicular communication and decentralized decision making is not readily supported.

AIMSUN is enhanced by a dynamic link library implemented in C++ which can be seamlessly integrated into the microscopic simulation environment (see Figure 4.5). In each simulation step, AIMSUN provides the current state of the simulation via its API Module to external applications, which may update the current state in an appropriate way. To use, process and manage this data, an object oriented data model is implemented, representing the current state of the simulation, e.g., vehicles' attributes such as current speed, route, and destination. This is the base for the implementation of functionality regarding dynamic traffic management, communication, and decentralized decision making.

In order to establish decentralized decision support, a modular software architecture has been developed that ensures manageability of the simulation framework. In Figure 4.7, the representation of an individual vehicle as an autonomous agent is depicted. Each vehicle has a navigation system, containing a graphic representation of the road network (GPS). The communication module (CommBox) allows reception and broadcasting of V2V and V2I messages. Such messages are the prerequisite for decentralized decision making in terms of grouping (Grouping App) and individual routing (Routing App) based on updated information by centralized traffic management. Since each vehicle has its own, self-developing navigation system, the Learning Application (Learning App) features continuous observation of the vehicle's anticipated and realized behavior and thus updates each vehicle's knowledge about the urban traffic network. From a centralized perspective, dynamic traffic management is supported by V2I-based travel time determination. Selected strategies are then disseminated to individual traffic participants in terms of weights that modify the attractiveness of specific turns.

AIMSUN Extended simulation results allow insights into the efficiency and applicability of the simulation framework. The communication models are crucial for agents' decision making, which usually implies ideal communication and availability of information. In sum, the consideration of surrounding buildings significantly affect dynamic traffic management applications in city road networks.

Communication

Communication interlinks the centralized and decentralized approaches. Figuratively, communication acts as a 'postal service' for information and the whole simulation can be executed on grounds of communication. Specific characteristics of urban areas are modeled for infrastructure and vehicular communication in order to guarantee the realistic collection and dissemination of decentralized information. The vehicles are equipped with on-board units (OBU), which support real-time information exchange and processing, communication with the traffic management center (TMC) and other vehicles. They process data and make autonomous decisions about routing and propose them as recommendations to the drivers. The TMC receives information from the vehicles, processes it and makes globally optimized decisions, which are returned back to the vehicles in the form of messages and signals of the traffic control infrastructure.

Multi-agent Model

The focus of this thesis is on agent-oriented simulation and the model of autonomous vehicle decisions in an urban traffic environment. The AIMSUN extended approach intends to integrate local decisions into global traffic management strategies. Traffic participants achieve their individual goals by forming groups and improving their knowledge about the road network, called 'learning'.

The data flows in the PLANETS architecture are demonstrated in Figure 4.8.

Status Quo

The AIMSUN Extended architecture is demonstrated in Figure 4.9 with its strengths in modeling the environment and its algorithmic simulation behavior. The PLANETS project ended in 2011 and continuing research was to add communication with the tool OMNeT++ and multi-agent models with JADE, as planned in Figure 4.5 of AIMSUN API. Some publications like [108] were done on the AIMSUN extended approach.

Since AIMSUN is a fully comprehensive, but a commercial traffic simulator, alternative traffic simulators should be considered for communication and agent educational research. One appealing alternative is the open-source traffic simulator SUMO.

AIMSUN Extended Discussion

Three requirements are fulfilled, only agent accuracy needs improvement with agent model extensions like the JADE middleware - see Section 4.3.4.

- **Agent Accuracy:** With AIMSUN Extended, agent-based modeling was only integrated via the API in C++ hard-code. It showed that it is possible, but an agent-based model which can cope with the communication like

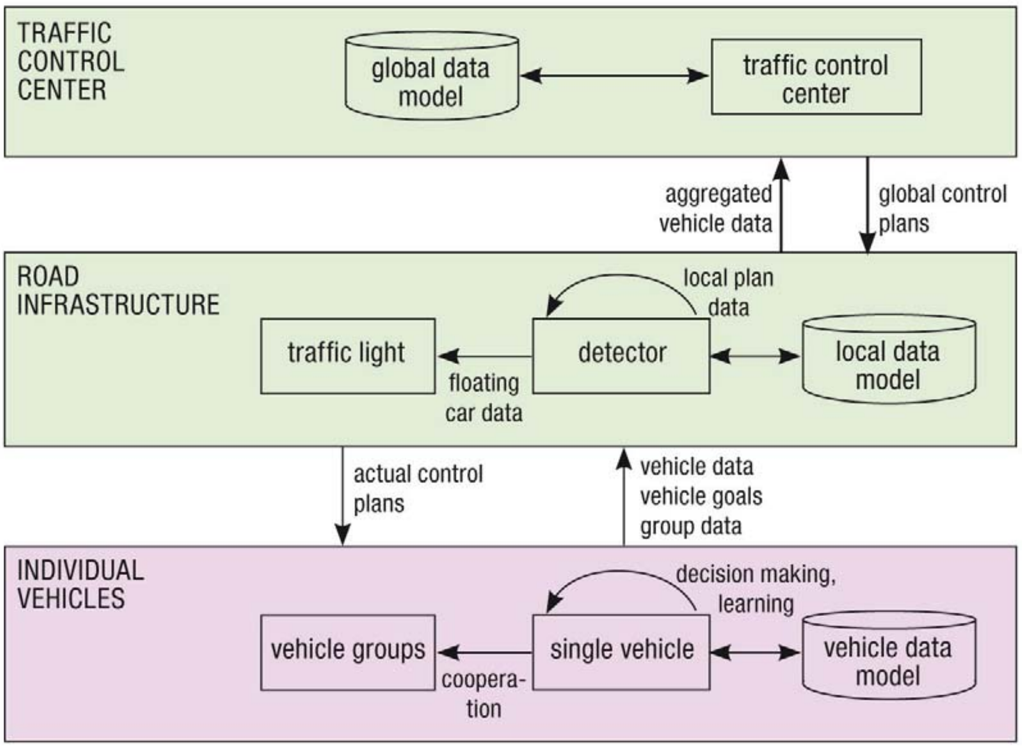


Figure 4.8: AIMSUN extended data model.

JADE would be very helpful. This is done with ATSim in Section 4.3.4. This would enrich the autonomy of vehicle agent behavior and decision making.

- **Integration with traffic control:** AIMSUN simulator with its extension combined the decentralized intelligent vehicle agents with the centralized simulated traffic control system (i.e., streets, intersections, traffic lights) with communication and data handling. Experiments were successful with different agent and environment control mechanisms. The integration of traffic control is a big plus of AIMSUN and its extensions.
- **Small-scale simulation:** With AIMSUN the degree of detail and accuracy is given. The computational performance slows down for agent coordination and decision making in the form of groups and routes, because these are calls via the API.
- **Realistic dynamics:** Using the AIMSUN traffic simulator with its extension, it is possible to simulate thousands of agent vehicles traveling through different pre-defined realistic road network scenarios at a certain density (Level of Service C-D) within the road network of Hanover, Germany.

  The simulator has all the necessary traffic models and algorithms integrated and provides realistic simulations.

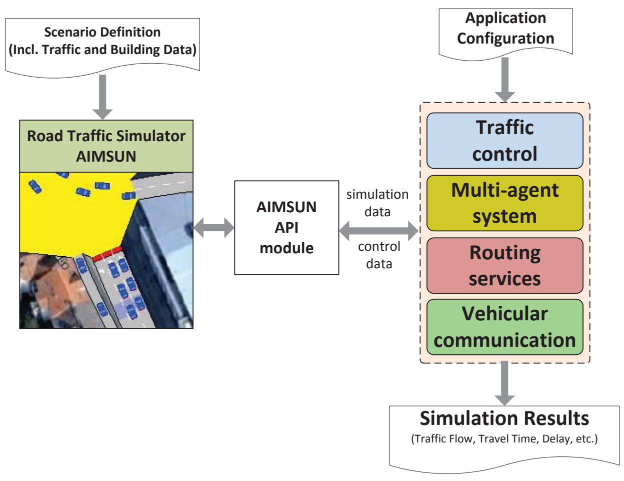


Figure 4.9: AIMSUN Extended Architecture.

The non-functional requirements are met because AIMSUN is a mature program, but with the extensions it is more limited, due to its commercial nature:

- **Functionality:** AIMSUN is very suitable, but due to its commercial character it is hard to extend with specialized communication and agent simulation. The models are correct and it runs on different systems although the PLANETS project used it mostly on Windows machines. Two and a half requirements are met, but agent and communication can be improved with external simulation input and performance. The accessibility is very good once it is licensed, although importing images can take some time. The trial version does not have all functionalities - especially since the Expert Edition provides access to the SDK.
- **Reliability:** AIMSUN is a very mature program and is recommended. The extensions are inserted as prototypes in order to show that simulations can be done on the same base. Fault tolerance and recoverability were not tested thoroughly, but appear to be good.
- **Usability:** AIMSUN has all environment and algorithm support, but interaction and individualization could be done better with external simulators rather than hard-wired via API. AIMSUN is intuitive and easy to learn. It provides extensive manuals for all details. The usability is simple for small networks, but lots of features can be added and changed and the user could get overwhelmed.
- **Efficiency:** No performance tests were made, but time and resource behavior seems to be okay for the realistic scenario of southern Hanover, Germany. With extensions in communication and agents it takes a lot more time and resources and simulations can take days for one peak traffic hour.
- **Maintainability:** AIMSUN comes with lots of features for analyzing, but it is only limitedly changeable with API where more is desired for the

purpose of this research. AIMSUN is stable and tested, and the extensions are tested, but can have bugs and performance problems.

- **Portability:** AIMSUN is not customizable. It is installable and complies with norms and conventions of portability. Another option is to replace it with the open-source microscopic traffic simulator SUMO (See Section 4.3.5).

All in all, AIMSUN is good for the environment and provided algorithms, but needs to be significantly extended with interaction and agent features. This is not easy because AIMSUN is commercial and not every method can be accessed easily.

4.3.4 ATSim: Agent-based Traffic Simulation System

This section is based on the CAEPIA [65] conference paper and a diploma work as part of this thesis project by Viet Hung `Chu` [64]. ATSim extends the transport simulation AIMSUN with JADE. A group-oriented driving method is implemented in ATSim.

There are various multi-agent grouping technologies applying general coordination and cooperation processes, e.g., `Levesque` [68], `Doran` [88], `Luck` [234], and `Wooldridge` [379], but no solutions fit to the traffic domain.

Methods of coordination and cooperation in MAS are considered, for instance, in `Consoli` [71], `Findler` [103], `Salazar` [302]. `Barrett et al.` [22] examine models for instant agent teamwork without considering communication and sensory aspects for an simplified application domain.

For that reason, at microscopic level various traffic scenarios with autonomous vehicles are provided with a model, simulation, and analysis. On the run-time side, AIMSUN [20] can model and simulate traffic scenarios, and integrate the environment with the JADE interpreter [35] for agent-based implementation. The combination results in an '**A**gent-based **T**raffic **Sim**ulation System (ATSim)' [65] and [64].

JADE is used and published for two platform combinations: First, JREP [138] is created as a novel integration of JADE and Repast Simphony that efficiently combines the macro and micro perspective with an interaction layer. Second, ATSim [65] represents another novel integration of the traffic simulation framework AIMSUN for modeling and simulating traffic scenarios and the multi-agent programming system JADE to implement the multi-agent behavior.

JREP focuses not only on the overall system behavior of the macro perspective, but also allows the micro perspective for local coordination and cooperation of individuals together with their interests, goals and communication. JREP [138] an environment where new agents register themselves including aspects of the scheduling of the agents and time synchronization of combined simulation systems. For modeling and simulating on the JREP platform combination, an agent-based airport scenario was proposed. In addition, a coin flip scenario was described for validating scalability and performance properties.

On a macro-level view ATSim supports global system throughput and on the individual vehicle micro-level preserves decentralization and openness. Hence, the infrastructure elements of the traffic domain such as traffic lights and/or vehicles can use the agent paradigm. Research on multi-agent traffic light coordination and cooperation has been done by`Bazzan` [26], whereas routing has been addressed by `Prothmann` [279]. However, decentralized vehicle agent coordination and cooperation is still open for investigation, which is considered with the group-oriented driving method in my works of [65] and [64].

Today's traffic simulation systems are widely used to analyze and manage traffic flows. Most of the research on traffic simulation systems [20, 57, 160] use mathematical models for modeling and simulating driving behavior, e.g., for car following and lane changing. The effect of communication and cooperation is missing in those models; as well as these models do not cover information, goal, and plan states of *autonomous vehicles*. In order to make the vehicles of a traffic simulation system communicative and to support flexible autonomous behavior, the paper ATSim [65] created a traffic simulation system with autonomous agents which introduces a decentralized group-oriented driving method, spontaneous team formation and conflict solutions.

Installation

In summary, the ATSim approach [65] extends the transport simulation AIMSUN with JADE. AIMSUN is used to design and run traffic scenarios, because it offers good support for modeling the complex road network infrastructure, elaborate vehicle behavior and control of traffic lights. JADE is used for adding individual agent behavior and communication between the agents to the AIMSUN traffic scenario. For connecting the two platforms the AIMSUN API was used, which allows external applications to access traffic objects. The AIMSUN API is only accessible with the programming languages Python and C/C++ and therefore CORBA was used as a middleware to provide the traffic objects in the programming language JAVA for the platform JADE.

As shown in Figure 4.10, ATSim is a composition of three main components: AIMSUN traffic simulation, the CORBA middleware and JADE agent simulation, which need to be installed. The connecting elements are the MAS Connector consisting of a Data-Reader-Writer, a Process Controller and a Data Converter on the traffic simulation side, which exchanges data with the AIMSUN API and on the other side the MAS Services and Agent Controller integrated with the Agent Container as the MAS Manager.

1. **AIMSUN Simulation API:** This contains a traffic network for urban traffic, the infrastructure elements like traffic lights and a simulator to make experiments. It executes simulation processes and communicates with external applications via the provided API.
2. **MAS connector:** This is necessary since vehicles in AIMSUN are created dynamically and controlled by agents of a MAS. Agents need information of vehicles and simulated models (e.g., highway, traffic light) for decid-

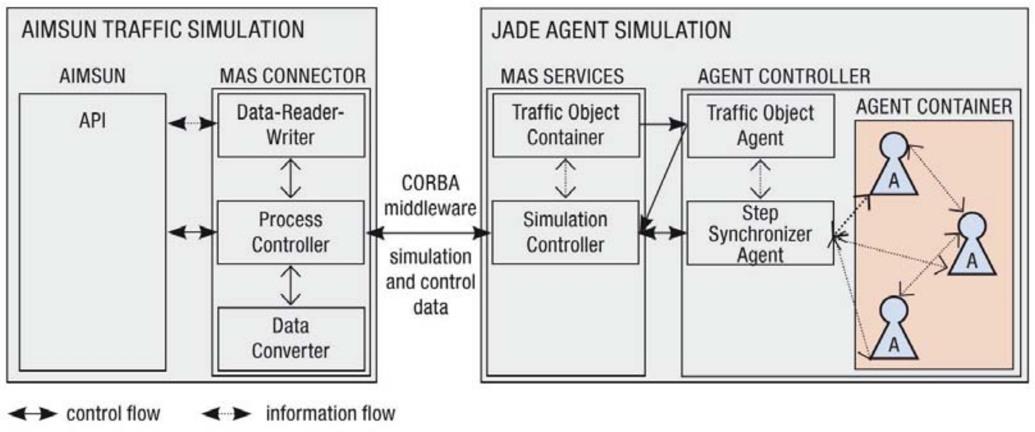


Figure 4.10: ATSim system architecture.

ing on their actions. MAS connector is used for exchanging information between AIMSUN and JADE and transforming the data with the data converter. The data converter transfers static and dynamic information into appropriate CORBA-objects and, in reverse, the control orders of agent vehicles into useful information for simulating traffic objects.

3. **CORBA middleware:** The CORBA Service is used for the exchange of information between the MAS connector and the services which require the representation of static and dynamic information as CORBA-objects, converted by the data converter of the MAS connector. CORBA stands for '**C**ommon **O**bject **R**equest **B**roker **A**rchitecture and specification'. For more information see `www.omg.org/oma`.
4. **MAS services:** This is an interface which allows JADE to communicate with traffic objects of the traffic simulation system. For controlling traffic objects like vehicles and traffic lights the interface provides services for creating and controlling agent life cycles. MAS services initialize the simulation model, create and delete agents, actualize the dynamic information of the agent, and arbitrate for a new simulation step.
5. **Agent Controller with container:** This is the main component of the MAS and provides the environment for managing agents and administrating the traffic objects. It consists of the simulation controller, the agent controller with traffic objects and a container with the agents. JADE agents which exist in agent containers can communicate.

Further and more detailed information is provided in the diploma thesis by `Chu` [64].

Simulation Scenario

ATSim is an extension of AIMSUN combined with communication and coordination abilities in JADE and implementation of group-oriented driving. By describing the architecture, all installations and connections are described.

The use of traffic simulation systems (TSS) [20, 57, 222, 332] for simulating traffic scenarios is widespread in the traffic management domain. A TSS provides an easy way to model and configure various traffic scenarios. However, for simulating behaviors of traffic participants hard-coded mathematical models are used to simulate vehicles. Thus, they lack communication with others and autonomous driving actions. Since communication and autonomy capacities of vehicles are the most important requirements of group-oriented driving, ATSim employs an **a**gent-based **t**raffic **sim**ulation system as a test-bed for group-oriented driving [65].

In a simulation scenario the new architecture of ATSim was tested on a traffic scenario with two intersections, shown in Figure 4.11. The group-oriented driving is presented in the scenario where vehicles with the same desired speed are put into one group. This method can solve conflicts because, a slower group blocks a faster group, it can react. In every group there is a group leader who coordinates the tasks and choses the lane for his group. Every group member has the individual freedom of choice as to whether or not they want to drive in the group lane. The vehicles try to follow the principle that the autonomic vehicle drives in that lane at its desired speed and does not block his follower.

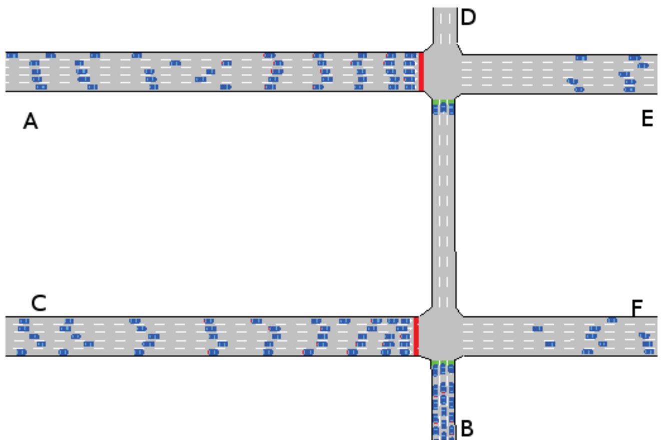


Figure 4.11: ATSim Traffic Scenario with two intersections created with AIMSUN.

The group-oriented traffic coordination solves the problem illustrated above in Figure 4.11 by proposing *Group-oriented driving* (GoD), a new autonomous co-operative driving method. In order to avoid and solve the conflict situations autonomous vehicles are able to perceive their environment, communicate, form groups, and co-ordinate their behaviors. The commercial traffic simulation platform AIMSUN of simulating driving behaviors servers as the reference to the implemented GoD method. Simulation experiments that suggest the method can reduce travel times and delays, while the overall benefit of the approach depends on the structure of the vehicle population.

The simulation scenario validates the scalability and performance of ATSim by means of simulation experiments. Compared to a simulation with standard traffic parameters, the results of speed and delays of each experiment show many advantages with the group-oriented driving method.

This simulation scenario describes three different types of vehicles: (1) fast, (2) medium and (3) cautious. The properties of the different class types are allegorical through desired speed, maximum delay, and acceleration. This test is generated twice with AIMSUN. First, data is collected with a standard driving setting before ATSim is used in a second run to control the group-oriented driving method. The following properties are designed to the three classes of vehicles:

- fast: desired speed $160km/h$, maximum delay $-8m/s^2$, acceleration $6m/s^2$;
- medium: desired speed $80km/h$, maximum delay $-8m/s^2$, acceleration $4m/s^2$;
- cautious: desired speed $60km/h$, maximum delay $-6m/s^2$, acceleration $2m/s^2$.

With the group-oriented driving method, fast vehicles can go at their desired speeds and improve delays and acceleration times, reaching their goals without blocking others. Therefore the delay of all vehicles is reduced, as seen in Figure 4.12.

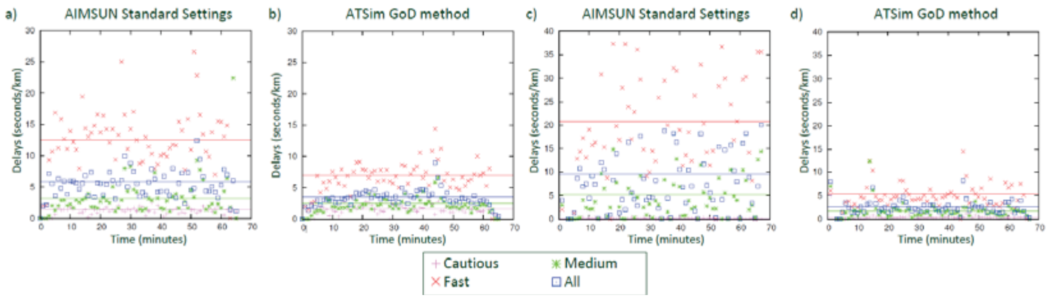


Figure 4.12: Simulation results of the group-oriented driving method.

Here, in Figure 4.12, (a) shows the results of manned vehicles in the standard simulation, while (b) expresses the results for *autonomous* vehicles with the group-oriented driving method. In (a), the medium class distribute its delays in the range from 0 to 23 s/km. The average delay of a medium class varies by 3 s/km (green horizontal line). Hence, the delay of medium manned vehicles deviates up to 20 s/km.

In comparison, when the group-oriented driving method is applied, the average delay is approximately 1 sec/km lower. It is remarkable that the maximum deviation of average delay decreases to only 5 seconds during the simulation. In consequence, deliveries of truck drivers can be estimated more accurately, become more reliable, and a logistics company would, e.g., be able to inform their customers more precisely of estimated delivery time frames. Looking at the fast class, there are similar results: they have better travel times in the autonomous simulation. For the cautious class the results are almost identical which is not surprising as cautious vehicles are the slowest and block other traffic participants rather than being blocked themselves.

In real traffic, the differences of vehicle properties in a group are small. Figure 4.12 show simulation results with slight variations in the parts (c) and (d) with maximum difference in the desired speed of -10 km/h for vehicles within one group, and in maximum delay and acceleration of $-0.5m/s^2$.

- Fast: desired speed $[150, 160]km/h$, maximum delay $[-7.5, -8]m/s^2$, acceleration $[5.5, 6]m/s^2$;
- Medium fast: desired speed $[80, 90]km/h$, maximum delay $[-7.5, -8]m/s^2$, acceleration $[3.5, 4]m/s^2$;
- Cautious: desired speed $[50, 60]km/h$, maximum delay $[-5.5, -6]m/s^2$ and acceleration $[2, 2.5]m/s^2$.

The speed of autonomous vehicles in one group is influenced by properties which have a negative effect. Especially the cautious – compared to manned cars – are more often delayed when autonomous vehicles are simulated due to harmonizations of speeds within a group and cooperative lane changing.

As a consequence, while forming autonomous vehicle groups, property differences among the group members should be minimized. The group-oriented driving method delivers in total better results in both tests presenting big differences *between* vehicle groups by means of desired speed, delay, and acceleration.

Status Quo

The ATSim architecture has problems with performance with bigger vehicle numbers. I.e., 1.2 seconds and 160 MB memory are necessary to calculate a simulation step in a traffic scenario with 2000 agents. For smaller scenarios ATSim is appropriate.

The results show that the group-oriented driving method gives the vehicles speed advantages.

A specific traffic scenario with the group-oriented driving method was considered. To determine whether the group-oriented driving method provides advantages over the compared standard driving method in situations when vehicles drive very close to each other in dense traffic requires deeper analysis. First results in evaluating the group-oriented driving method have been promising in 2011, supported by the diploma thesis of `Chu` [64] and the conference paper [65]. Mainly the work with ATSim finished in 2011, mainly because of the bottleneck of two simulations: a traffic and agent simulation connected with the CORBA middleware. The approach of Streetworld in Section 4.3.1 was thereafter followed up and other work has been done with AIMSUN itself and its extensions, as discussed in Section 4.3.2.

Future work will examine and compare different group coordination strategies for determining group life time, and strategies of initiating and abandoning groups. Initial investigations have been done with richer vehicle models including strategy learning, and considering interaction of agents and infrastructure components (such as traffic lights). First results that study local and group decision-making strategies based on agent-oriented data-mining and reinforcement learning are described in `Fiosins et al.` [106]. At the same time, different traffic management scenarios are considered including optimizing throughput at individual and subsequent intersections, or co-operative group-based routing of platoons in Section 4.3.2. A further line of research are communication strategies (with whom to exchange what information) as conducted in `Fiosins et al.` [109]. To this end, previous work done in the context of agent-based supply chain monitoring by `Guo and Müller` [147] could be revisited.

In addition, the plan is to enhance the simulation of autonomous vehicle agents by enabling them to communicate with traffic light agents in order to increase traffic flows in other, more complex traffic scenarios like intersections [65] (see Figure 4.11).

Further development to combine AIMSUN with agent interpreter JADE was done in [65] and used in [140], but both simulations were found to have a synchronization bottleneck and work stopped in 2011.

ATSim Discussion

Four requirements are fulfilled, the agent accuracy can improve with BDI agent model extensions like Jason see next Section 4.3.5, and realistic dynamics can implement realistic data.

- **Agent Accuracy:** With ATSim, agent-based modeling was integrated via the JADE middleware which is especially good for communication with the FIPA standard. Simple agents with a driving and conflict solving method were implemented in an agent-based model which can cope with communication. The autonomy of vehicle agent behavior and decision making needs to be implemented - no BDI corpus is given like in Jason, described in Section B.4.
- **Integration with traffic control:** Just as in AIMSUN Extended, the simulator had an extension for agent models with a strong focus on communication. A simulated traffic control system (i.e., streets, intersections, traffic lights) is nicely covered with AIMSUN traffic simulator. The group-oriented driving method was successful with conflict handling and individual agent driving.
- **Small-scale simulation:** With ATSim the necessary degree of detail, the same as with AIMSUN, and accuracy is given for agent coordination and decision making in the form of groups, which handle conflicts and drive individually. The computational performance is limited to small scenarios, but it is within the requirements.
- **Realistic dynamics:** Using the AIMSUN traffic simulator with its JADE extension, the aim of simulating up to a thousand of agent vehicles trav-

eling through different pre-defined artificial road network scenarios is fulfilled. Here the bottleneck can cause problems with larger scenarios.

The simulator has all the necessary traffic models and algorithms integrated and provides realistic simulations.

ATSim is a prototype and does not meet all non-functional requirements. AIMSUN and JADE are mature simulation tools, but the extension of AIMSUN with JADE and CORBA is limited due to its commercial nature.

- **Functionality:** ATSim is suitable, correct, interoperable, and meets the constant to the above-mentioned requirements. It might have problems with accessibility because it is not public, only research work at the Technical University of Clausthal.
- **Reliability:** The three components of ATSim, AIMSUN, CORBA and JADE, are mature tools, which ensures fault tolerance and recoverability.
- **Usability:** ATSim is explained very comprehensively. In order to learn and use it, knowledge of the three components is needed.
- **Efficiency:** ATSim has problems with time and resource behavior, but works for small simulations.
- **Maintainability:** ATSim is just a prototype and therefore the criteria of being analyzable, changeable, and stable have not been tested enough to make a judgment.
- **Portability:** ATSim is a good approach, but does not fully meet the criteria of being customizable, installable, compliant with norms and conventions of portability, and being replaceable.

For larger numbers of vehicles ATSim is relatively slow in simulating the traffic model due to synchronizing two simulations. For example, the simulation of one simulation step in the case of 2000 vehicles demands up to 1.2 seconds and 160 megabyte memory. Thus, at the current stage, ATSim is suitable for simulating small to medium traffic models with a limited number of agents (simulation at the microscopic level). The good news is that both time and memory grow approximately linearly with the number of agents, which may suggest that the size of the simulation can be increased by a linear factor by using more powerful hardware or distributed computing resources. This preliminary hypothesis needs further experiments for confirmation.

4.3.5 Multi-Agent Traffic Interface (MATI)

In this section, an alternative approach is described, which combines existing traffic simulators with agent-based (simulation) platforms. This main thesis approach is described in more detail in Chapter 6. The MATI research focuses on coupling existing simulation platforms with the MAS view point. The goal is to provide efficient, modular, and extendable simulation platform for traffic systems that, in particular, supports the microscopic (and possibly the mesoscopic) perspective of MAS and their coordination. It may be used for other things than traffic environments too and could become a MAS simulation platform with different environments and tools. MATI is primarily used for small-scale traffic agent simulation to show dynamic vehicle group formation which supports, on principle, the activity and communication behavior of traffic participants, especially the vehicles.

Note that the use of the term 'interface' in MATI is slightly misleading: MATI is more than an interface: It is a modular multi-platform framework built on the EIS standard developed in the thesis of `Behrens` [31], which can be instantiated to use variable traffic simulators together with different agent platforms. Thus, simulations are not bound to any specific interpreter or environment.

MATI provides the interfaces for the extended EIS framework to plug in various environments, interpreters, and tools, which can be combined or used as stand-alone components. Modules can be replaced by various implementations to simulate different scenarios.

So far, MATI has been instantiated to combine the SUMO [215] environment with the interpreters Jason [49] and JaCaMo[2] and Simple Tool [3] and HDF 5 [4] as illustrated in the MATI package at the bottom of Figure 4.13, but is not limited to this. The gray wild-card characters indicate that other tools, environments or interpreters used as plug-ins.

Thus, MATI offers a framework in which to simulate general environments e.g., demand modeling and multi-agent simulations, as well as tools to analyze the output generated by the modules. The MATI simulation uses real-world data for creating the agent-traffic system. The simulation scenarios are described in chapter 6.

MATI is a tool which allows the addition of multi-agent properties to add to traffic simulation. The simulator is responsible for all the environment characteristics and provides MATI constantly with actual outside world information which the agents understand as perceptions. Furthermore, the simulator realizes actions of agents which are associated to vehicles, such as route changes. In addition, the simulator is in charge of the rudimentary driving behavior of vehicles like keeping the proper distance. MATI extends the driving behavior of vehicles with the BDI standard of the multi-agent architecture. Thus, vehicles can plan goals, execute them and change them. Furthermore, MATI provides communication and organization structures of vehicles with each other via agents.

A MATI simulation runs sequentially. The agents act in the same time frame. Only if all the agents have executed their reasoning cycle, do all the agents get the signal to start a new one. The process in the simulator is also coupled to this tact. In every round, before the reasoning cycle of the agents,

[2] *JaCaMo*, available at `http://jacamo.sourceforge.net`, combines three well-established MAS technologies: *Jason*, available at `http://jason.sourceforge.net`[49], *CArtAgO*, available at `http://cartago.sourceforge.net`[292] and *Moise*, available at `http://moise.sourceforge.net`[173].

[3] attached in the Appendix (self created)

[4] *HDF 5* is available at `https://www.hdfgroup.org/HDF5/`

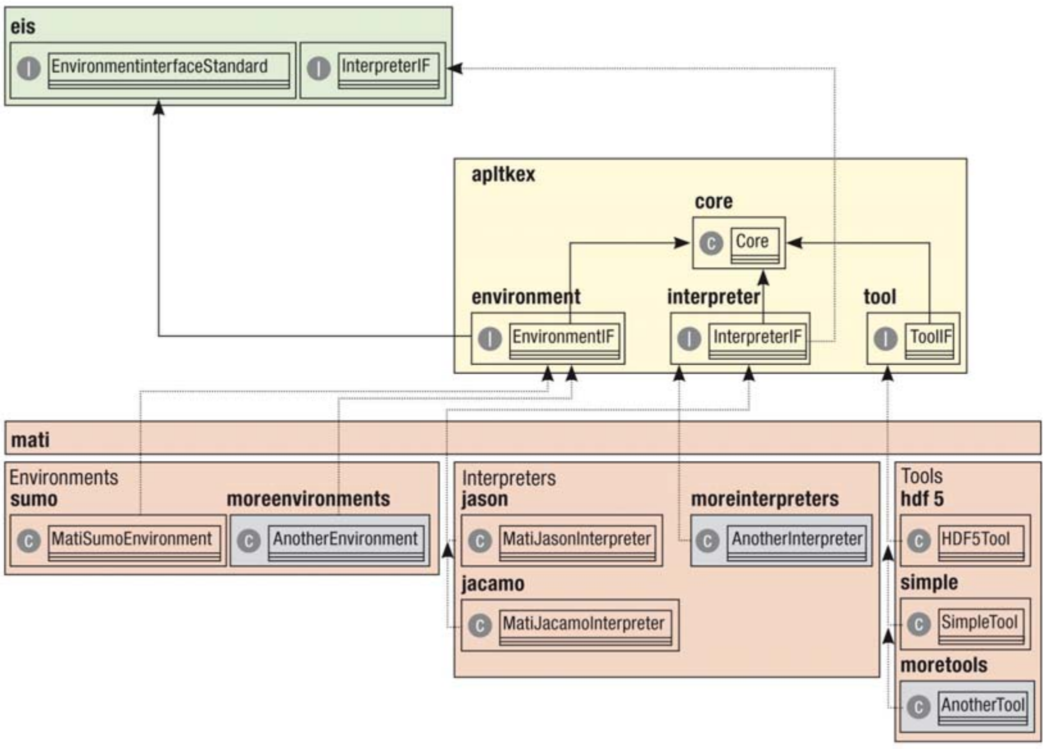


Figure 4.13: MATI Packages.

the simulator checks whether the amount of vehicles has changed. After that, the agents are created or deleted. Thereafter the percepts of the agents are actualized with new information from the simulator.

During the development of MATI, the focus was not on scalability, therefore no clear statements about the scalability of the system can be offered, besides the performance test provided in Chapter 4.

Unfortunately, while there is a considerable body of work on agent-based traffic modeling and simulation, many different platforms are used and a standard has not yet been defined. Research is progressing as summarized and discussed in `Chen and Cheng` [62] and `Bazzan and Klügl` [29]. To maximize the efficiency of dynamic and decentralized urban traffic systems, traffic components are expected to cooperate.

The motivation behind MATI is that there is no suitable platform that simplifies the interconnection of BDI agent platforms with traffic simulation environments and analyzing tools. MATI is required in order to abstract from the underlying tools and offer simple accessibility through an API upon which programmers can implement test scenarios. Agents, for instance, which control the behavior of vehicles that are running on the platform, must be able to communicate and interact with each other in order to show group effects. The platform must therefore offer instruments for communication and coordination between agents.

JaCaMo is one possible alternative (next to JADE agents) to the pure Jason interpreter used in MATI main. JaCaMo combines the Jason interpreter with CArtAgO environment and the organizational tool Moise+ through the artifact concept introduced by `Boissier, Bordini, Hübner and Ricci` [45, 173].

For example, the JaCaMo agent, called 'init_JaCaMoAgent.asl', perceives artifacts of types and paths and concatenates with the type and to add an identification (Id) with its name. It creates an artifact with its path and focuses on the Id to perceive the new artifact. A log file returns "test" and then it can start acting in the environment depending on its path.

```
!init.

+!init
    <-
        .my_name(MyName);
        for(perceptArtifact(ATyp,APath)){
            .concat(ATyp,"_",MyName,AGlobalName);
            makeArtifact(AGlobalName,APath,[],Id);
            focus(Id);
            addPerceptArtifact(ATyp,Id);
            log("test");
        };
        //!!start;
        .
```

Listing 4.1: JaCaMo Environment Perception.

This way the agents can perceive a vehicle with its path. They can follow another vehicle's route and therefore change their own route to the newly adopted target which is triggered.

```
perceptArtifact("vehicle",
 "mati.interpreters.jacamo.artifacts.VehicleArtifact").

{ include("init_JaCaMoAgent.asl") }

+!start
    <-
        log("change");
        changeTarget("e3");
        .

+trigger
    <-
        log("trigger");
        .

+p(N)
    <-
        log(N);
        .
+speed(Speed)
    <-
        log("speed: ",Speed);
        .
```

Listing 4.2: JaCaMo Agent Reaction.

Here, the initiating vehicle 'VehT' is telling its route to others to follow him:

```
+route(R)
 <-
 .println("My route: ",R);
 .list(R);
 .println("Hello vehR! Don't you prefer to drive my route? :)");
 .send(vehR,achieve,changeRoute(R));
 .
```

Listing 4.3: JaCaMo Example Route.

Then, the following vehicles 'VehR' change their plans and adopt the new route/goal.

```
/* Plan Library */
+route(R) /* event trigger */
 <-
 .println("My Route: ",R);
 .
+!changeRoute(R)
 <-
 .println("Ok, I am about to change my route...");
 changeRoute(R);
 .
```

Listing 4.4: JaCaMo Example Change.

This was done in a small fork scenario starting with one lane and splitting into two different paths with 10 agents. The same concept is used for the Green Wave scenario where a group leader requests that members follow its route. JaCaMo restricts implementation with their given body to describe the combination of interpreter, environment and organization. On the other hand it is an integrated concept of all important AEIO agent features (agent, environment, interaction and organization), whereas the communication feature is included in the agent interpreter.

Since MAS can easily contain thousands of agents, the information exchange must be efficient and without much overhead. The platform must also link agents with entities in the simulation environment. Most simulators like AIMSUN (compare to Sections 4.3.2 and 4.3.4) offer an API to acquire updates of things that have changed in the world and also allow the manipulation of objects during runtime.

MATI must invoke these calls on behalf of the agent. These calls may be sent over a network connection if the simulator demands it. Usually interfaces are programmed for a specific version of the simulator. If the simulator changes, it is necessary to refit the API calls. MATI should run on major operating systems so that existing simulators, interpreters and tools can be connected to the platform.

A similar approach to the MATI idea by `Rossetti et al.` [299] influences drivers' behavior by improving driver modeling with BDI agents. An agent-based traffic simulation was conducted in `Balmer et al.` [16] that also distinguishes between environments, interpreters and tools with different name conventions but for macroscopic dimensions. The approach of `Bazzan et al.` [26] is more on the mesoscopic perspective, where traffic signals are grouped for learning agents in order to improve the global outcome.

Installation

The extensive **installation** for MATI with the technical details is described in Appendix C.1.

Simulation Scenario

Section 4.3.5 demonstrates the necessary steps for setting up a traffic simulation scenario using MATI. For this example, Jason is used as the agent language interpreter and SUMO as the simulation environment.

To connect the environment, the simulator ideally provides a Java interface, to enabling it to act as a MATI environment. SUMO has a TraCI4J[5] interface to translate SUMO C++ into Java (in ATSim in Section 4.3.4 a similar translation needed to be done from AIMSUN C++ into Java). With a TraCI4J Java interface the simulator can be controlled and simulation steps can be triggered manually. Simulation world entities such as vehicles should be controlled by agents and need to provide properties such as route and speed.

Likewise, a MATI interpreter is connected, provided that the Multi-agent System possesses a Java interface. The agent cycles need to be initiated manually. Perceptions deriving from the environment need to be transferred into the belief base of an agent and actions intercepted to modify changes in the environment, i.e., speed changes.

The linking between agents and entities is an essential part of MATI (described in detail in Subsection 4.3.5). This includes the perception of observable properties from the environment, committing actions and the communication between agents.

MATI uses a JSON[6]-File to describe a process. It contains a core and scenario specification (see Figure 4.14). The *core specification* describes common settings for the core like the initial step delay, the number of steps to skip and debugging functionalities. The scenario itself consists of environment and interpreter files which include the simulator SUMO and the interpreter Jason. Both programs are bundled as JAR-files and the relative path is provided in the JSON-file. Parameters are passed to the programs by defining a key-value-pair within the environment section. Hypothetically, this allows any simulator and any interpreter to be integrated into MATI.

In SUMO, a city scenario is created through a combination of single road sections (edges), intersections (junctions), traffic lights, roundabouts and multi-lane tracks. The traffic-related part of a map is described in a *SUMO networkfile* [7]. The network file is provided as a key-value-pair in the JSON file. Thus, SUMO is initialized with the proper scenario.

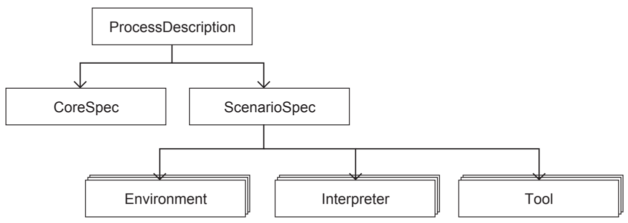


Figure 4.14: Process Description of a Scenario.

[5] *TraCI4J* is available at http://traci4j.sourceforge.net

[6] JavaScript Object Notation: open text-based format to transmit data objects as attribute value pairs

[7] http://sumo-sim.org/userdoc/Networks/

Agents, Artifacts, and Entities To connect entities and agents, MATI has to know which vehicles are currently running in the simulation. The simulator must offer an interface to query such information. SUMO uses a small server that uses a protocol called TraCI to steer and observe objects over a network connection.

In every step, MATI gets a list of active entities and determines which vehicles have been added to or removed from the map. For every new vehicle it adds a new agent and connects this agent with the ID of the entity.

MATI queries all entities in every round and informs the agent about changes. The agent can then react to these events and perform proper actions in the environment. Agents are software components. When new agents are added to the net, MATI can either assign an agent to the entity or get this information from the simulator.

In SUMO, a vehicle is defined in a *routes file*. This file contains the vehicles, their routes and other properties that define them. The XML-format was augmented by associating a new attribute, *agentType*, with the vehicle object. This attribute provides a path to an agent source file. MATI uses this information to link new vehicles to the defined agent.

Artifacts and agents is a concept by `Ricci et al.` [291, 292] from the environment seen by agents. They are objects or entities which are dynamically constructed resources and tools of the working environment. They contain triggers for operations which an agent can execute like a traffic light or a clock. The agent must observe the artifact before it can execute an operation. Agents can focus on many artifacts and therefore gain more perceptions and execute operations.

In MATI artifacts are used to connect observable environment properties with the agent. All vehicle agents create a new artifact themselves during initialization that that connects to an entity within the simulator. The agent is then focused on a single vehicle and receives property updates from it. The agents' operations work only on the connected vehicle. An environment is represented by one or more artifacts. An agent can focus on relevant artifacts to determine which environment it is working in. The vehicle artifact is used by all agents to perceive the current position on the section of the road network and the speed. Observable properties will trigger an event in the agent code if they change. Using this method, the agent can update its belief base, which is needed in order to select proper plans.

Evaluation and Performance In this section, the results of a preliminary experimental evaluation of Jason agents in SUMO are reported. Modularity often comes at the cost of overhead. Data must be passed through many interfaces to get from source to destination. The number of agents within a scenario will affect the time needed to execute one simulation cycle. Also, the complexity of an agent is an important matter. A scenario may run flawlessly with thousands of simple agents but might drop in performance if more sophisticated agents are used. Therefore, two types of MAS scenarios with different agent / interaction

Component	Selection
CPU	Intel(R) Xeon(TM) E312xx (Sandy Bridge) CPU @ 2.40GHz
Memory	8.0 GB
OS	Windows Server 2008 R2 Standard SP1 64-Bit
SUMO Version	0.17.0
Jason Version	1.3.9
AplTk Version	0.1

Table 4.2: Specification of the Test Machines.

complexities were tested, Simple *"Hello world"* agents and more computation-intensive *"Fibonacci"* agents, which are described in Appendix C.3.3.

Experimentation is carried out on a virtual cluster node with a fixed number of CPUs and fixed memory size on equal machines (software equipment). Experiments were run under equal conditions. Table 4.2 shows the system specifications.

Status Quo

MATI is a collection of components linked together by AplTkEx. Components of the MATI package from Figure C.1 are presented in the following. The gray packages are future work. In the current instantiation of MATI, the traffic simulator SUMO maintains the environment context; it supplies MATI with current environmental data, which the agents process as percepts. MATI allows Vehicle-to-X communication and coordination structures by modeling vehicles and infrastructure components as agents. MATI uses the *AgentSpeak* Language with Jason as the interpreter to encapsulate individual agents. Environments represent the external state and define how agents connect to and update the state, for which SUMO is used. The MATI framework is the tool that evaluates the internal state of an agent and environments to fulfill a distinct and useful purpose is the MATI framework.

MatiSimpleTool: MatiSimpleTool is a first implementation for the tool component. So far, its functionality is restricted to providing basic statistical configuration data such as the number of agents for each simulation step.

MatiSumoEnvironment: The traffic simulator SUMO can act as an environment through the MatiSumoEnvironment binding with diverse agent interpreters. For example, if a new vehicle enters the simulation, the interpreter is informed about it and can interlink an agent to the vehicle to control its behavior. This connection is done with the help of TraCI4J[8].

MatiJasonInterpreter: The MatiJasonInterpreter (see Fig. C.1) allows the interaction of Jason-based agents within one MATI simulation.

[8] *TraCI4J* is available at http://traci4j.sourceforge.net

MatiJacamoInterpreter: The MatiJacamoInterpreter is an extension of the MatiJasonInterpreter and permits the agents to use environmental CArtAgO elements and therefore also to use Moise agent organizational structures.

MATI Discussion

MATI completely fulfills the requirements, as is addressed in the following.

- **Agent Accuracy:** With MATI, individual decisions of drivers, traffic controllers and all the actors that are part of the traffic system are elegantly rendered by the concept of artifacts and agents. Agents are modeled with detailed, rich BDI behaviors for individual entities and are very accurate. Vehicle groups are implemented practically in Chapter 6. The simulation example illustrates how MATI simplifies connecting BDI agent platforms with traffic environments including evaluation and analyzing tools. Simple API access is provided and linking agents with entities in the simulation environment is explained in Section 4.3.5.
- **Integration with traffic control:** MATI enables experimentation with different agent and environment control mechanisms thanks to its modular nature. The MATI system has many components, consisting of the environment, the interpreter and the tools which are exchangeable and combinable. There are different dependencies between them which need to be implemented by the designer. The system components and the entire system act non-deterministically and are not completely synchronized; processes in the system run concurrently. Thus, there is competition for the system resources. MATI is as open as a "Lego" system: new components can be added to MATI, even those that were unknown at the time of development. Also, new dependencies can develop between components new in the system during runtime, or existing dependencies can break. The self-adaptive structure and behavior of the system can be changed by MATI.
- **Small-scale simulation:** The MATI research focuses on individual and small group coordination mechanisms compared to the default global traffic behavior. This can be quite complex depending on the degree of detail and the specific domain knowledge of traffic, agents and mathematical analyzing tools. MATI can simulate thousands of entities driven by agents with a high degree of detail and accuracy. The computational performance is good enough for small-scale simulation, as shown by the performance test. With the help of Jason, MATI also connects to the CArthAgO concept: building a bridge with environment artifacts steered by agents for modeling and engineering working environments. This is how it deals with the complexity of different domains, so that the traffic and agent simulation do not need to be run simultaneously like in Section 4.3.4. Instead, the agents observe the traffic simulation and are triggered to take actions when needed. MOISE is good for the organizational concept and

is implemented in MATI with JaCaMo for modeling and simulation. In any case, the performance could be improved in future versions of MATI. MASON B.5 serves as an example due to very good performance.

- **Realistic dynamics:** The aim is to simulate up to a thousand agent vehicles traveling through different pre-defined artificial and realistic road network scenarios at a certain density (Level of Service C-D). This assures the possible application to academic comparisons of models and realistic traffic in cities. This is done in MATI in the following chapter 6 and works very well. In the evaluation Chapter 7, 5275 agents are used in different artificial and realistic scenarios of Hanover, Germany with collected data. The simulator SUMO includes different traffic models and scenarios are created with the required density of vehicles.

Regarding the non-functional requirements, MATI is very good in functionality and portability, but, as for the other requirements, it would need refactoring to improve reliability, usability and efficiency:

- **Functionality:** With Java interfaces, MATI is platform independent and suitable for the purpose of forming vehicle groups in urban traffic. It can run on major operating systems so that existing simulators, interpreters and tools can be connected to the platform. MATI runs correctly and is consistent with the functional requirements analyzed above. MATI is a prototype development investigated and established by the Technical University of Clausthal and available upon request. Further investigations are needed to test the communication and interaction strategies.
- **Reliability:** MATI is an alpha framework and is mature enough to test vehicle groups and other scenarios prototypically. Although mature components are used, the MATI framework would need to be redesigned for reliable use with fault tolerance and recoverability.
- **Usability:** MATI needs some time to understand - also due to the use of the different components: the specifications of the environment (here SUMO), the agent interpreter (JaCaMo or Jason) and the tools (SimpleTool or HDF5). For example, JASON, as mentioned in Section B.4, is not very easy to learn, but powerful for designing accurate agents.
- **Efficiency:** The efficiency of MATI was tested in the performance test illustrated above in Section 4.3.5. The efficiency is sufficient for scenarios to run in an acceptable time and resources are used depending on the complexity of the agent. Here, only individual parameters are measured; in the future the influence of different parameters on one another could be investigated.
- **Maintainability:** MATI is difficult to analyze due to all the influencing components, but it has the advantage of being changeable to any environment and different agent types as well as analyzing tools. MATI runs stably with small scale scenarios with 5275 agents, but on the artificial grid scenario it runs into its limits (see Chapter 7).

- **Portability:** The MATI platform is modular and extendable. Interpreter, environment and tool can be replaced by others of the same kind. At present it uses the Jason Interpreter, the SUMO environment and HDF5 Tool (and Simple Tool) to analyze the simulation. JaCaMo was also tried as an interpreter, but for the remains of this thesis only Jason is used due to performance reasons and for reasons of simplification.

A short overview of related work by `Behrens` [31] was given in Section 2.3.6 with AplTk and EIS as a base for the main simulation component of this thesis. Extending AplTk and using EIS, MATI is able to connect different environments with various interpreters and tools whilst making the code re-usable. Therefore, a multi-agent framework including the BDI-like architecture for modular multi-platform simulations is provided. All research questions with (traffic) agents can run in small-scale simulations and are analyzed in Chapter 7. In the simulation scenario, it is demonstrated how integration and programming of interpreter and environments in MATI works is demonstrated. Technically, MATI is a combination of SUMO and Jason and insight on agents and artifacts as resources are given. The discussion shows that the requirements are met. MATI is a plug-in platform for connecting (traffic) environments with agent platforms which can run as just the agent platform or with arbitrary environments and tools. Different types of agents have been tested for coordination within the environment and different scenarios.

The modular MATI framework combines environments, here, traffic simulations, with interpreters, interlinking individual agent behavior with the environment technically employed by influencing autonomous vehicles and different analyzing and evaluation tools. For a more detailed overview of the scaling of the framework, more complex and bigger scenarios with different agent types must be run, which were not required for the purpose of this thesis. Mixed agent type scenarios should deliver more detailed information about the work load and the micro and macro perspective of scaling. Thus, information about each method and data call can be retrieved, so statistics about the execution time of different agents, actions e.g., mail checking, action run and their distribution in the agent cycle, can be created. With this information, a macroscopic view of the scenario and the difference between the behavior of local agent groups and the whole simulation can be formed. In later experiments, more complex definitions of agents were needed, as well as the grid structure of the scenarios and data storing within the agents in order to create a more in-depth statement.

MATI performance tests show that it is suitable for medium-sized scenarios with up to 5,500 simple agents. Thus, traffic flows of a small city can be calculated. It can be used for smaller or bigger scenarios depending on the computational performance. The simulation runs centrally on a single machine. Therefore, it is not possible to run a bigger simulation on several computers and distribute the calculation throughout a computer cluster without significant changes. For the performance test with MATI a single work station with the parameters in Table 4.3 was used.

Specification	**Selection**
CPU	Intel(R) Core(TM) i7-3770 CPU @ 3.4GHz
Memory	16 GB
OS	Windows 7 64-Bit
SUMO Version	0.17.0
JAVA JRE	1.7.0
Jason Version	1.3.9
AplTk Version	0.1

Table 4.3: Specifications for Performance Test.

Two metrics were measured: (1) the amount of agents per simulation cycle and (2) the amount of simulation cycles per second. The amount of agents in a scenario is not limited. The computer only calculates longer for one simulation cycle. In one of those cycles the perceptions are actualized and the agent cycle runs through for each agent. Thus, each agent also has its own cycle in which it reacts to perception, executes its plans and chooses its next action. The next action is put into a queue, and in the next run of the simulation cycle, it is performed. The second metric is more significant for the duration of a simulation with a given amount of simulation steps. The correlation between the amount of agents and cycles per second is almost linear: with 3000 agents and six cycles per second requires therefore 6000 agents and 3 cycles per second. The cycles are calculated discretely (there are no half cycles), therefore the amount of cycles varies on the boundaries until it stabilizes at a constant value.

SUMO is a step-based simulator. If SUMO is triggered from the outside, it executes all new actions and repaints the picture. This means that, with 3 cycles per second, the picture is updated three times per second (3 FPS). That seems like a slow rate, but it is sufficient for most simulations. The amount of cycles is defined in the simulation. In the case of SUMO, it is done in the corresponding configuration file. Generally, the amount of agents entering the traffic net corresponds to the amount of agents exiting. The average fluctuation range represents only a few hundreds of agents and influences the cycles per second insignificantly. With 7000 cycles and 15 cycles per second the simulation takes 7 minutes.

4.4 Summary

As traffic becomes more complex and coupled with new technologies, modeling and simulation must also become more sophisticated. The requirements are outlined for a decentralized, bottom-up approach compared to the classical top-down paradigm. Thus, more dynamic and direct traffic-dependent solutions can be simulated and evaluated - modeling traffic scenarios with agents is a suitable method. But, for this, controlling mechanisms or information need to be distributed and communicated, which raises the need for new coordination protocols.

When comparing agent-oriented simulation platforms for cooperative traffic, Jason and JADE seem the most appropriate for cooperative traffic due to their strengths in agent accuracy, especially with the BDI paradigm provided in Jason and possible with JADEX and FIPA communication standards. That is why they are used for further research and simulation. First, Jason as a stand-alone agent interpreter was used in Streetworld, described in Section 4.3.1. Getting started with Jason is difficult, but once the knowledge is gained it is a powerful BDI agent simulation. Jason is a good choice for developing scenarios with accurate and interacting agents. AIMSUN, as the traffic environment with its strengths in environment and mathematical models as a stand-alone, was extended, but did not give satisfying individual and interaction results. Next, the middleware JADE in combination with the traffic simulator AIMSUN, called ATSim, was created, which had a disadvantage with the bottleneck of two simulations of data transfer between two simulations. Finally, the MATI simulation framework was developed and is used for vehicle groups in this thesis. It is further described in Chapter 6.

ATSim and MATI are both sophisticated and, in principal, cover all aspects. These two combinations are very advanced and connected with an external interface. Thus, the programming effort and domain knowledge is relatively high in comparison to only agent-oriented simulation for cooperative traffic, described in the comparison in Section 4.2. Other problems of the combination of platforms are performance quality and scalability fall-off, but this does not contradict the requirements for small-scale simulation and reasonable times. The advantage of the combination of tools is that environment, algorithms, interaction and individualization are covered and used with all aspects. This is the key for the following dynamic vehicle groups in urban traffic in Chapter 6.

τόν ἧττω λὸγον κρείττω ποιεῖν - the weaker cause could be made the stronger
Protagoras of Abdera ca. 490 BC-420 BC, pre-Socratic Greek philospopher

Chapter 5

A Model for Vehicle Group Formation

In chapter 3, it has been pointed out that other related approaches have mainly concentrated either on how to make traffic management more dynamic or on agent-based approaches to static top-down traffic control. According to `Vasirani` [363], so far little research has been dedicated to integrating decentralized dynamic coordination methods into traffic control. This thesis aims to fill this gap. For implementing those methods, a platform which combines traffic control with agent simulation is required, which integrates the modeling into one framework. This is discussed in Chapter 4.

This chapter studies the model for vehicle group formation from the decentralized perspective of autonomous vehicle decisions in an urban traffic environment. The approach of this thesis should improve traffic flow, benefiting individual vehicles, but indirectly the overall traffic network.

This Chapter is organized into eight main sections of the group life cycle reference model illustrated in Figure 5.3. My own publications ([175, 176]) influenced this chapter, which presents a layered architecture for co-operative traffic management, considering the top-down traffic management features and focusing on a decentralized bottom-up agent-based approach in order to bring both perspectives together with the help of vehicle groups. This makes it possible to model motivation (see Section 5.4.2), individual behavior (see Section 5.2), and methods and structure for grouped traffic participants (see 5.4) as well as local traffic context (see Section 5.3.1). In Section 5.5.2, realistic V2X communication models and protocols are used to simulate information exchange between vehicles and the TMC.

5.1 The Conceptual Model

The first step is the general development of a conceptional model (cf. [206] p. 47). The most important phase is the qualitative description of the impact of measurements within the model. In a multi-agent model,the representation of the behaviors of an individual agent or agent types, and the parts of the environment and their dynamics is relatively abstract. The general methodology of the model is a metaphor which describes the point of view for the modeling. Thus, the model specifies the allowable experiments. Depending on these experiments, the validity can be investigated if the model corresponds sufficiently with the original. The hypothetical ideal for simulation experiments is the model base which is valid for all experiments, although the construction of such a model is very rarely practical. Within a concrete experimentation frame, a simple or lumped model can be constructed which is valid for all experiments. More information about the hierarchy of models, experimentation frames and their validity can be found in `Zeigler` [389] and other standard works about simulation techniques.

Dynamic Traffic is a large socio-technical system and consists of numerous subsystems, which are often developed and provided independently by different organizational bodies. Thus, exhaustive testing or even verification of the functionalities is limited and not feasible with all parts. In addition, humans develop, use, and modify the system in manifold ways. Notably, those interactions are often done in or with the subsystems mentioned.

With growing on-board intelligence in vehicles and Vehicle-to-X communication capabilities, it is expected that future traffic systems will have more dynamic control and a decentralized architecture. Today's centrally managed traffic management centers are likely to be replaced by system configurations where traffic entities such as signals or vehicles will be self-managed and autonomous. Communication, coordination and negotiation between these nodes will occur. Standards for communication between autonomous vehicles and their coordination through dynamic traffic management need to be established.

Decentralization is recognized as the tendency to replace hierarchy through planning by self-dependent organization of economic activities and their coordination in traffic oriented negotiation processes (refer to `Bullinger` [24] p.1).

Decentralization has no standards itself, but it is useful in small sub-systems for bottom-up coordination including vehicular communication). Grouping an optimization process between two extremes: completely centralized or decentralized ([24] p.3). Two characteristics of decentralized organization structures are: process orientation and cooperation. Process orientation increases the reaction rate, reduces the information paths, and optimizes of flows. Cooperation needs communication and coordination as a base for increasing performance. Communication provides an important potential of the setup for the coordination, distribution, and use of specific and overall information ([24] p.4).

Whereas the process orientation is handled by traffic management, the focus is on the autonomous vehicles with communication, coordination and cooperation in order to reduce complexity for the organizational framework and divide the system into manageable sub-systems, which are easier and more flexible to control. Group-orientation is the key element where strict structure is retrenched and responsibility is based on competence in small work packages, which are delegated to groups. Working in groups needs to be practiced for good performance ([24] p.6) and henceforth is touched, but is out of scope for this thesis.

Decentralized or hierarchical coordination protocols and new dynamic control mechanisms are needed for the paradigm change from centralized to decentralized traffic management. These need to be simulated and analyzed before being put into real world practice. A suitable paradigm is needed for modeling and simulating these kinds of systems, for which Multi-Agent Systems (MAS) [382] are a promising candidate. In order to fulfill real world requirements, autonomous agents need to adapt and perform tasks in a dynamic environment such as the traffic system. This needs to be considered when modeling the agent and the environment.

In this thesis, the focus is on the integration of centralized and decentralized approaches in the form of groups, where global control strategies are provided from TMCs, as is customary, but local freedom is delegated to the traffic control elements or participants. Whereas traffic signal control is much researched and one of the key factors of urban traffic control, the key actors are the traffic participants like vehicles, bicycles and pedestrians. Therefore, the local decisions of the traffic participants need to respect the corresponding traffic regulations. Each traffic participant makes his or her decisions autonomously based on his or her interests and available information; this can be best automatized in vehicles. Therefore, vehicles are the research object representing traffic participants in cooperative traffic. Observing these decisions or influencing them e.g., by means of dynamic traffic signs or through sanctions and incentives while preserving the autonomy of the vehicles is another key to optimizing traffic control [112]. There is a large amount of research on individual decisions of traffic participants [106, 107], but there is little work on integrating decentralized participant decisions with a centralized control approach [139, 140].

Completely decentralized control with only vehicles' decisions as the rule on roads is not feasible due to the local perspective and self-interested, egoistic actions, which would cause poor overall social benefit [313]. One hypothesis of this thesis is that integrating decentralized decisions into a centralized approach using the dependency of groups allows traffic participants to make autonomous decisions, which can make traffic control more efficient and user-friendly.

5.1.1 A-E-I-O-S Categories

In order to decompose the complex behavior of autonomous vehicle groups in cooperative urban traffic, this thesis separates the components into four main categories shown in Figure 5.1: agents, environment, interaction and organization. They are combined for simulation in order to analyze varying measurement inputs and evaluate different scenarios. The links symbolize the complete connection of all separate components. Those categories are viewed from the perspective of the domain of Multi-agent Systems, but with a strong focus on traffic behavior.

These categories represent the organizational aspects of this thesis.

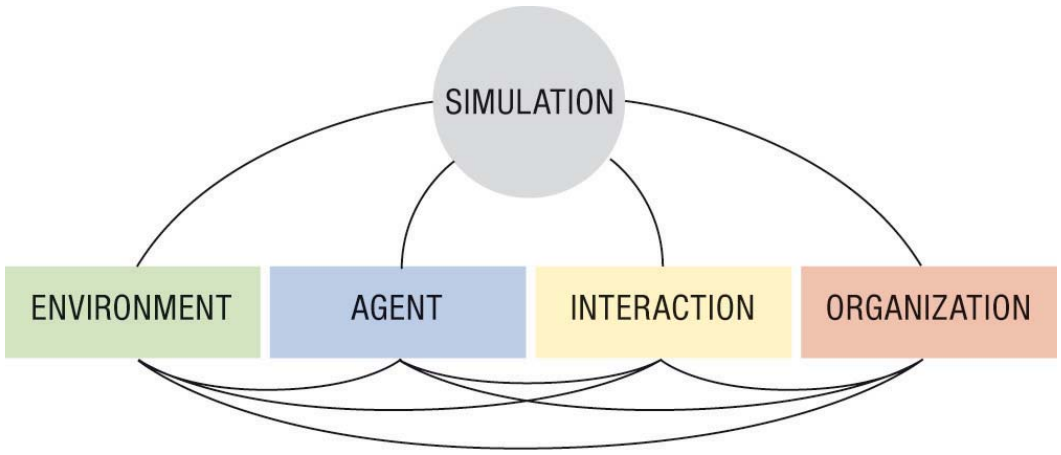


Figure 5.1: **S**imulation fully interconnected with **E**nvironment, **A**gents, **I**nteraction, and **O**rganization.

Agents

The agent represents a participant of an environment represented in a computational system. The participant could stand for a technical subsystem or a human being in a virtual simulation. Humans are an integral part. They continuously interact with others and with technical systems like agents, requesting a service from an agent or an agent organization and using it, like autonomous vehicles. For that, an order should specify what has to be achieved, but the detailed plan of how to achieve the goal is made by the agent by enabling a plan or behavioral selection. For the scope of this thesis, autonomous vehicles incorporate the agent design and humans are not considered.

An agent is an entity observing and acting on an environment. For this they are equipped with sensors and interaction ability to gather information about the world, then they can interpret the data for their actual state and can create an output to change it. Due to different agent architectures, they are often illustrated as a black box: receiving input with their sensors, processing it and then, with the effectors, providing output to the environment. Agents have the following properties: they are reactive, social, proactive, and real-time compliant, as described in detail in Section 5.2.

Environment

Firstly, the environment is the context where many individual participants are situated physically in the real world. The environment has the following characteristics: it provides a static and dynamic infrastructure, it is an open system without boundaries but can provide a collection of scenes for coordinated activities, and it is a real world-compliant base for simulation, which is elaborated on in Section 5.3.1. Secondly, the environment is associated with traffic management and provides organizational units like traffic lights to foster coordination and regulate the interaction of autonomous, heterogeneous traffic participants like the agents beyond their physical and technical constraints. Traffic management can be seen as an institution that regulates the agents' behavior for balancing between the global and local interests. Thus, traffic management can establish and sustain certain notions of stability. Organization supports the coordination processes. This is discussed from the agent perspective in Section 5.3.2.

Interaction

Interaction includes direct V2X communication with different standards (KQML, FIPA, ETSI, Car2Car) of communication and indirect communication, depending on the behavior of the agents and respecting each other. Often the ability to communicate is integrated in agent functionalities. Interactions, however, are a key factor for controlling and balancing the global system with the interdependent participants in the environment. Changes in the environment, addition or removal of (vehicle) agents, behavioral modifications of traffic participants or new strategic TM goals can be communicated and a homeostatic mechanism to balance various influences and effects can be put into place by traffic management, so that a stable equilibrium state is maintained for security, comfort, and traffic throughput. Communication models are often integrated in the agent's design, and the remainder is explained in Section 5.3.1.

Organization

Organization is defined as a structure and is, together with institutions, an important governance element. Organizations facilitate the agent grouping, structure, and collaboration within the environment. The coordination of the environment such as Traffic Management Control is assisted by organizations, institutions with norms, and trust. There are two types of coordination: cooperation, in which non-antagonistic agents pursue a common goal, or competition, in which agents follow up on their different, partly opposing aims. This thesis only focuses on cooperative urban traffic and designs and implements autonomous vehicle groups. Usually, organization is embedded in the social layer of the agents, and should be connected to the Traffic Management environment. The organization of autonomous vehicle groups is described in Section 5.3.2.

Simulation

A simulation is an abstracted model of the environment in which many participants are situated virtually. For simulation, all the other components, agents, environment, interaction and organization, are represented in mathematical models, protocols and algorithms illustrated in Figure 5.1. The simulation is the execution base for the group formation models of this thesis and Chapter 4 is dedicated to a detailed evaluation of agent-based traffic simulation.

5.1.2 Functional Characterization

Functional relationships exist between the components of the traffic vehicle agent system. They define the flow of control in an agent, mapping its perceptions into action in the environment.

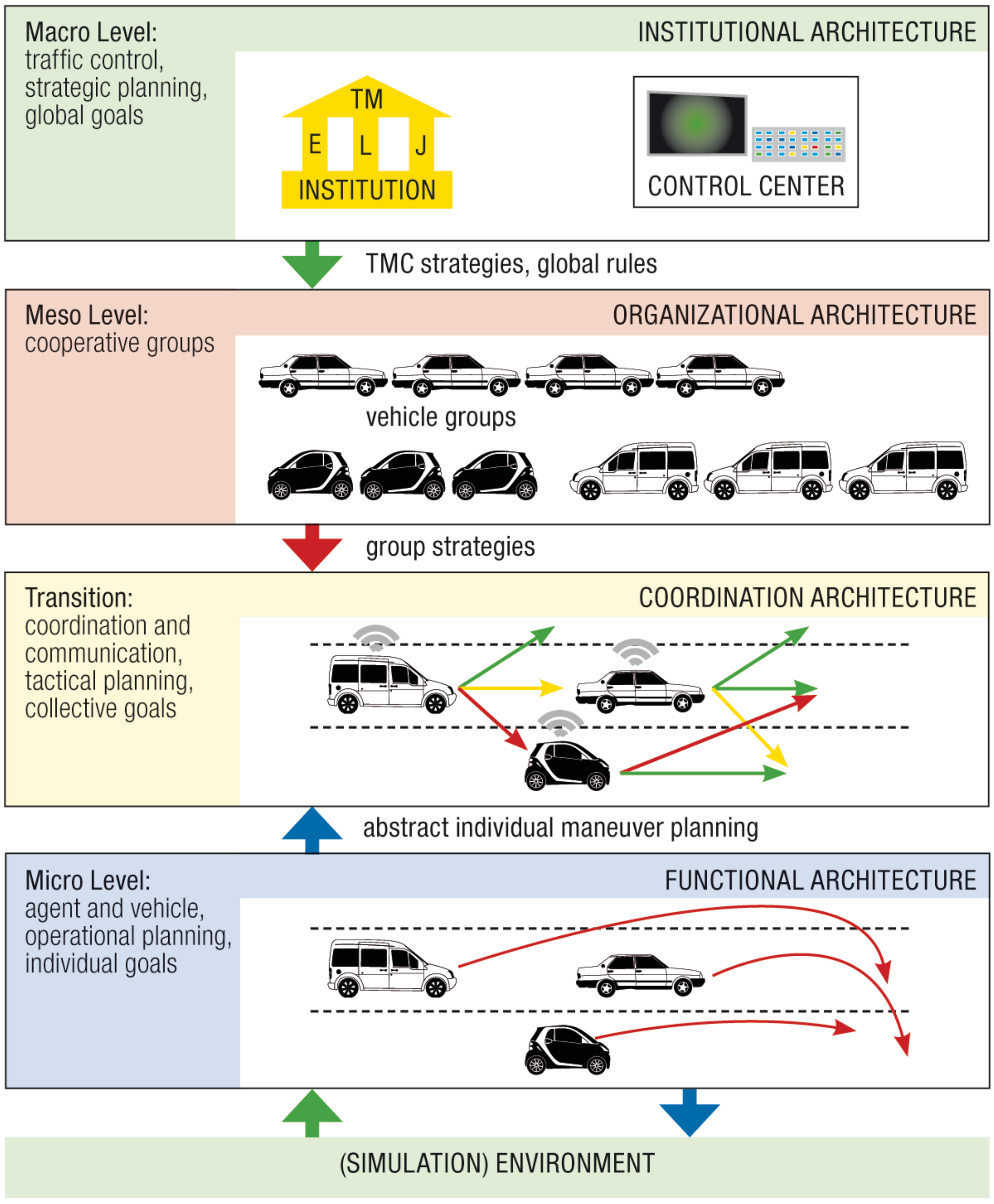

Macro Level:
traffic control, strategic planning, global goals
INSTITUTIONAL ARCHITECTURE
TM
E
L
J
INSTITUTION
CONTROL CENTER
TMC strategies, global rules
Meso Level:
cooperative groups
ORGANIZATIONAL ARCHITECTURE
vehicle groups
group strategies
Transition:
coordination and communication, tactical planning, collective goals
COORDINATION ARCHITECTURE
abstract individual maneuver planning
Micro Level:
agent and vehicle, operational planning, individual goals
FUNCTIONAL ARCHITECTURE
(SIMULATION) ENVIRONMENT

Figure 5.2: Levels of Architectures (derived from Sanderson [303], [304]).

The Micro Level: Agent and Vehicle

The micro level describes the autonomous vehicle in the form of an agent and embodies the combination of a mechatronic layer and an executive layer with its individual goals, based on a utility function.

Due to the focus of this thesis being on the social model and cooperating vehicle groups, the individual agent on the micro level consists of existing agent models described on the mechatronic layer and the executive layer. Further information for robotic and individual models can be found in Baber et al. [12], Stiller et al. (cf. [329] and my own ideas and related work in [64, 106, 107, 111, 137, 175, 176, 253].

Initially, the agent is blank regarding environment information, but adapts to environment geometry, which refers to the mechatronic layer in the AAOL agent architecture in Figure 5.4. The mechatronic layer couples the mechanical and electronic properties of an agent, provides sensor data and controls actuators.

Transition: Coordination and Communication

The transition between the micro and meso levels describes functions and realizes the distances and algorithms and the execution of groups.

In the AAOL agent architecture, the transition incorporates the Individual Context Layer for the decision-making process, including the existence of multiple agents in coordination as well as in communication. Those decisions are based on the environment information derived from the Executive Layer. The ICL reflects the individual world state depending on the plans and, finally, goals of an agent for controlling its behavior. Here, feedback control like the MAPE cycle approach [201] can be implemented.

The Meso Level: Groups

On the meso level of Figure 5.2, coordination happens. It is either distributed from the bottom up or organized centrally, top down. Group membership is a flag within the agent or a set in the coordination instance.

Multiple agents act in the same environment. Thus, negotiating is done in order to reach joint goals and plans and common beliefs are communicated to the members. Coordination and cooperation takes place in the social context layer, which correlates to the meso level, where i.e., resource conflicts solved.

The following models and concepts of vehicle groups 5.5, the weighting function 5.5, the group algorithm 2, group conflicts 5.5.1 and global coordination 5.5.1 were developed in collaboration with Hung Chu, who also described grouping concepts in his diploma thesis [64], and I depicted them in publications [139, 140, 141].

The joint goals of the BCC component are executed in the meso level, where a cooperative plan defines the tasks for each member. If the cooperative plan is accepted, the agent is committed to fulfilling its tasks. Accordingly, the agent extracts its group goal from the tasks. In every simulation step, a member of a group gets a task from the group leader. This task coordinates the agents in order to avoid a potential group conflict. Depending on the actual state of the agent, the agent can decide whether the task needs to be executed. For example, if an agent does not want to be blocked by slow driving agents, when it gets the information about this potential conflict, it needs to try to change lanes but, through global coordination, all members are assigned to group lanes. Therefore, the conflicted agent gets the task to drive in the group lane.

The Macro Level: Institutions and Traffic Management

Regulating and controlling social goals without remotely commanding individuals is the task of traffic management. Institutions can provide the structural organization of separation of powers, i.e., the executive power in form of police, legislative power to create regulations and norms and judicatory power to decide when conflicts arise. Traffic Management usually is an institutional body of a city for regulating the urban network and infrastructure with the aim to balance the traffic load and throughput.

For this thesis, one traffic state with fixed traffic control is assumed in order to study decentralized group effects on the traffic scenario. For future work, a feedback loop in form of providing green phase guarantees to groups could be included, as well as scenarios with different traffic state settings.

Group Life Cycle

The Group Life Cycle in Figure 5.3 is based on the AEIOS categories and identifies elements that can be influenced directly. As a counterexample, *trust* is an aspect of group culture which cannot be influenced directly. Instead, specific behaviors of norms, underlying values and assumptions can create or destroy trust.

Repeating information flows can be detected. In the field of inter-communication between single process areas, no established standards exist due to different organizational implementations depending on the goals of strategic decisions. However, sub-processes communicate and interact. The quality of communication is improved through the concept of crossing interactions within Group-Life-Cycle Management.

Beside the main processes in Figure 5.3 of group planning, group formation, and group operation (which is not specified and aspect of simulation because group work needs to be practiced), there are established process fields which are not always well-defined and assigned. Those connected processes in the cycle are analyzed and prepared as a reference model. The model helps new groups get off to a good start. Therefore, this reference model provides structure and organizes the following group components.

5.2 Individual Control

Basics about agents were described in the Background Chapter 2.3.1.

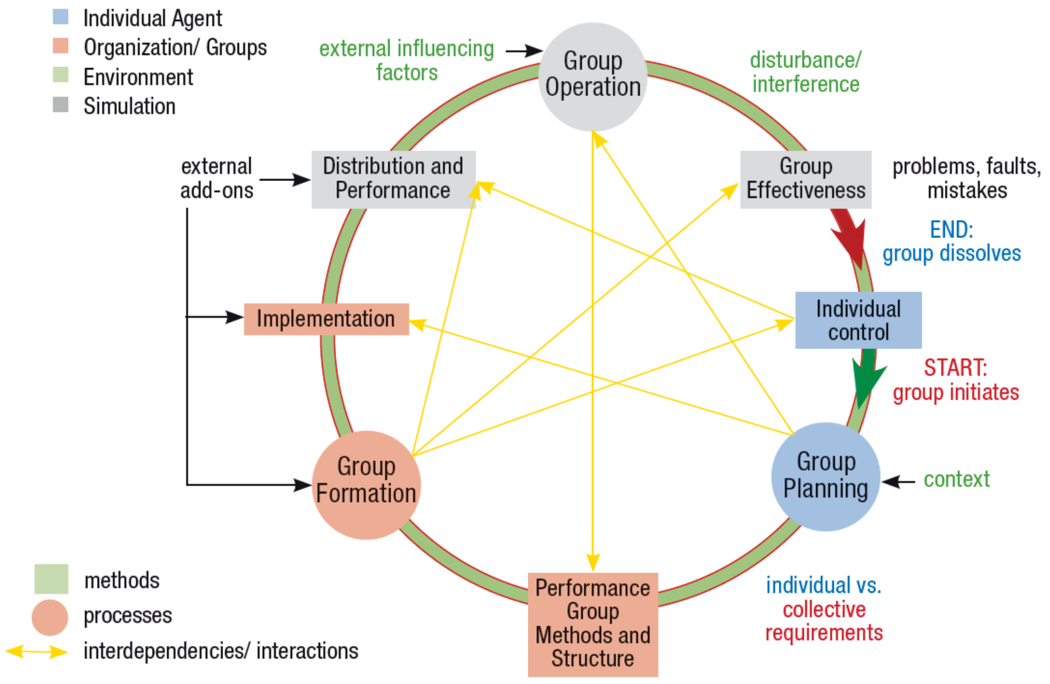


Figure 5.3: Reference Model: Group Life Cycle (adapted from Hackman [150] and Sundstrom et al. [333]).

The design for vehicle agents needs to fulfill the general requirements above (see Section 5.1) and particularly the following for autonomous vehicle groups:

- **planning:** the ability focuses on the short-term (tactical) and long-term (strategic) plans for reaching their goal. The agent should be able to create a local as well as a cooperative plan with several agents as members. For example, an agent cannot change lanes when there is the danger of collision with another agent. But the agent can make a plan to change to the desired lane in future time steps, possibly taking others into account.
- **negotiation:** the ability to interact and communicate is required to negotiate about a cooperative plan. The agent gets an invitation to take part in a cooperative plan. The agent should be able to accept or deny, but also to make an alternative offer. In this thesis, negotiation is only rudimentary, that is, for accepting or denying an offer.
- **cooperation:** the ability to do joint actions and follow a joint plan is cooperation. The agent needs to be benevolent. If necessary for the joint plan, the agent needs to relinquish its individual plan.

A layered architecture for social agents like the meta model and conceptual AAOL Agent architecture [175, 176] fulfills these requirements. The reasons it was chosen as the general architecture for this thesis are the advantages of modularity and the possibility to adapt to changes in design and context, which creates, on one hand, enough foundation for analytical modeling and specification of the traffic system. On the other hand it is broad enough for developing reasonably large agent environments. The disadvantages are the incapability of dividing the architecture into separate layers due to goals interfering with each other, and the difficulty of choosing an agent design caused by a highly distributed system of limitations and suppression, resulting in a rather low flexibility during run time. A hybrid architecture that is a combination of reactive and proactive behaviors, like the InteRRap architecture proposed by `Müller` ([253] and [256]), provides a good base for general abstractions and concepts.

`Müller` ([253] p.45) defines an agent using three interacting control and knowledge layers:

- *behavior-based* are routine tasks on the micro level
- *local planning* are local tasks and goal-directed behavior on the meso level
- *cooperative planning* is reasoning about other agents and cooperation on the macro level

These layers combine reactive reasoning (fast and flexible to environment changes) and deliberative reasoning (planning actions and goal-orientation), and interact with each other through negotiation. Therefore, the concept of layering is a powerful technique for integrating various functionalities and different levels of abstraction.

In contrast to `Müller`'s [253, 256] original architecture and the three levels of architecture presented in Figure 5.2, derived from `Sanderson` [304], however, the layers in `Huhn et al.` [176] are extended by the lower levels, dividing the world model in more detail. The AAOL agent architecture has been influenced by the InteRRaP agent architecture. `Huhn et al.`[176] propose an *agent architecture* - see Figure 5.4 - with four layers. From the bottom to the top, these are: the *Mechatronic Layer* (ML), the *Execution Layer*(EL), the *Individual Context Layer*(ICL), and the *Social Context Layer*(SCL). Depending on their goals, agents perform single actions or sequenced plans with individual consideration of the environment. For joint goals and joint plans, the view of the environment can be integrated and the information pertaining to negotiated goals and plans and the social world state is spread among multiple agents.

One authority controls each layer with an information flow for planning in the bottom-up direction and for execution in the top-down direction. In case this authority is unable to execute a task, it hands over the case to an upper layer. For example, as the EL detects an unresolvable problem, the ICL receives a request with the task. Once again, if the task is unsolvable individually and more than one agent is required, the ICL activates the top layer, SCL, to negotiate with others. Once a joint solution has been found, the actions are planned for the individual on the ICL, and can be performed on the EL depending on the mechanical or electronic properties of an agent. Note that this layered architecture was considered most promising to tackle agent behavior, but in the end was not operationalized with MATI.

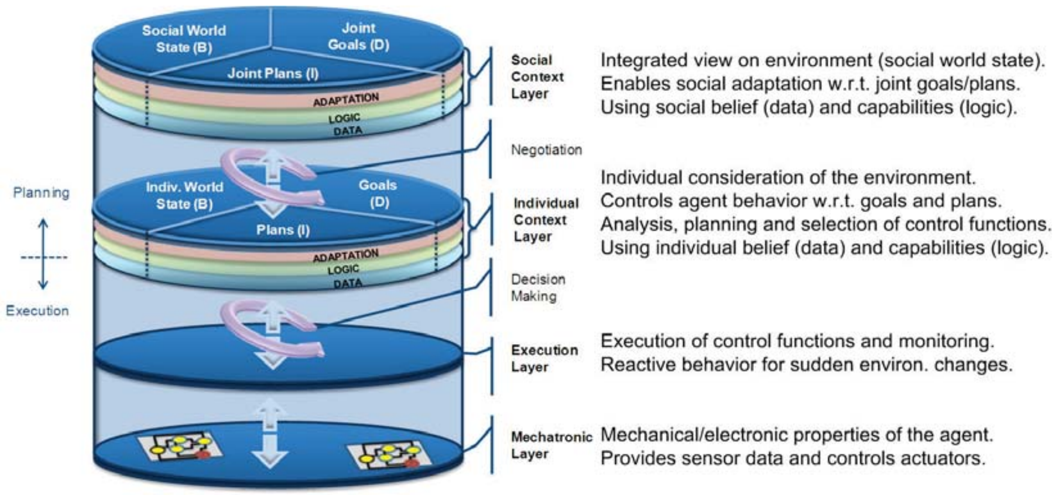

Social World State (B)
Joint Goals (D)
Joint Plans (I)
ADAPTATION
LOGIC
DATA
Social Context Layer
Integrated view on environment (social world state).
Enables social adaptation w.r.t. joint goals/plans.
Using social belief (data) and capabilities (logic).
Negotiation
Planning
Execution
Indiv. World State (B)
Goals (D)
Plans (I)
Individual Context Layer
Individual consideration of the environment.
Controls agent behavior w.r.t. goals and plans.
Analysis, planning and selection of control functions.
Using individual belief (data) and capabilities (logic).
Decision Making
Execution Layer
Execution of control functions and monitoring.
Reactive behavior for sudden environ. changes.
Mechatronic Layer
Mechanical/electronic properties of the agent.
Provides sensor data and controls actuators.

Figure 5.4: AAOL Agent Architecture (from Huhn et al. [176] p. 3/8).

5.2.1 Vehicle Agents

According to Papageorgiou [270], a lot of in-vehicle systems, also called automated vehicles, are emerging e.g., ACC. This includes collision warning, driver supervision, fully automated vehicles like Google car, lane keeping, active green driving etc. These technologies mainly improve safety and convenience, but only a few emerging vehicle automation and communication systems (VACS) have direct traffic flow implications. Cooperative systems are connected vehicles, also called VII, where mobile vehicles are the mobile sensors and have a direct link to traffic management e.g., CACC.

Papageorgiou [270] asserts that intelligent vehicles alone could lead to an unintelligent traffic flow and cause a reduction of capacity due to the following reasons:

- ACC with a long gap
- sluggish acceleration
- conservative lane change or merge assistance
- underutilized dedicated lanes
- inefficient lane assignment
- uncoordinated route advice

Why vehicles as agents? Multi-agent simulations are especially suitable for research on social objects: every vehicle (usually in combination with the driver to make decisions) acts autonomously and, moreover, decides only on the base of its local knowledge. Every change which a vehicle can execute within its environment only has a local effect. Despite simple behaviors (moving forward, right, left and rarely backwards) through self-organization complex and well-coordinated behaviors and patterns like vehicle groups emerge. Further characteristics like the existence of variable structures (vehicles can stop and additional ones arrive within the traffic network) show that the representation of a vehicle traffic network is a very complex modeling task.

Thus, the vehicle-driver combination is represented by agents. Within this thesis, the vehicles are autonomous and represent the combination of users driving through the traffic network and the vehicle properties like acceleration and deceleration. To indicate where a vehicle agent currently is positioned and where it is moving to, it sends out two different messages: exploring its neighborhood and sending its intention [110]. A vehicle agent drives its route from origin to destination. The life cycle of this agent also contains, besides exploring and the intention of group formation, functionality for updating its state, which contains information such as its origin and destination, the current segment it is driving on, and vehicle properties. If so configured, it also automatically selects a route from the solutions provided by the group leader. The vehicle agent offers two services to other agents or external users. First, the set of routing solutions, from its origin to its destination, can be requested. Second, a user can manually select one of these solutions to commit to. In this case the automatic intention selection is disabled.

5.2.2 Execution Layer

Additionally, on top of the micro level, the basic execution with control functions and monitoring are incorporated in a transition which is represented between the micro level and the meso level in Figure 5.2. In this execution layer, the reactive behavior of an agent to sudden environment changes is described.

The general structure of a traffic agent is adapted from the InteRRaP control architecture ([253] p. 53) by Chu's Traffic Object Agent ([64] p. 80), illustrated in Figure 5.5.

The general structure of Chu's Traffic Object Agent consists of three levels (refer to Figure 5.5): a 'world interface (WI)', 'Behavior Control Component (BCC)' and 'Plan Control Component (PCC)', where each level has its specific role and possesses an individual knowledge base - its own database on its level. The world interface has arrows, retrieving its information from sensors and communication and sending information by actions and communication to the environment. The arrow in the world model indicates the knowledge abstraction, where a lot of information is stored in the lower knowledge base and aggregated data relevant to the corresponding level is stored in the higher database. The task of the knowledge base is to store the knowledge of an agent depending on its level structure. Its access is organized incrementally. The objects of the upper level are constructed by the objects of the lower level. For example, all static and dynamic traffic objects are saved in the world model on the lowest level. The behavior, i.e., *DriveToTheLeft* of the BCC Level includes all dynamic objects (the other agents on the left lane) and the static objects (the left lane of the street) as component parts of the WI level and its structure. Thus, a plan of the PCC level is based on the behavior of the BCC level. Likewise, the InteRRap structure of the control flow is important for the distribution of the execution rights of its components. If one component is not able to reach

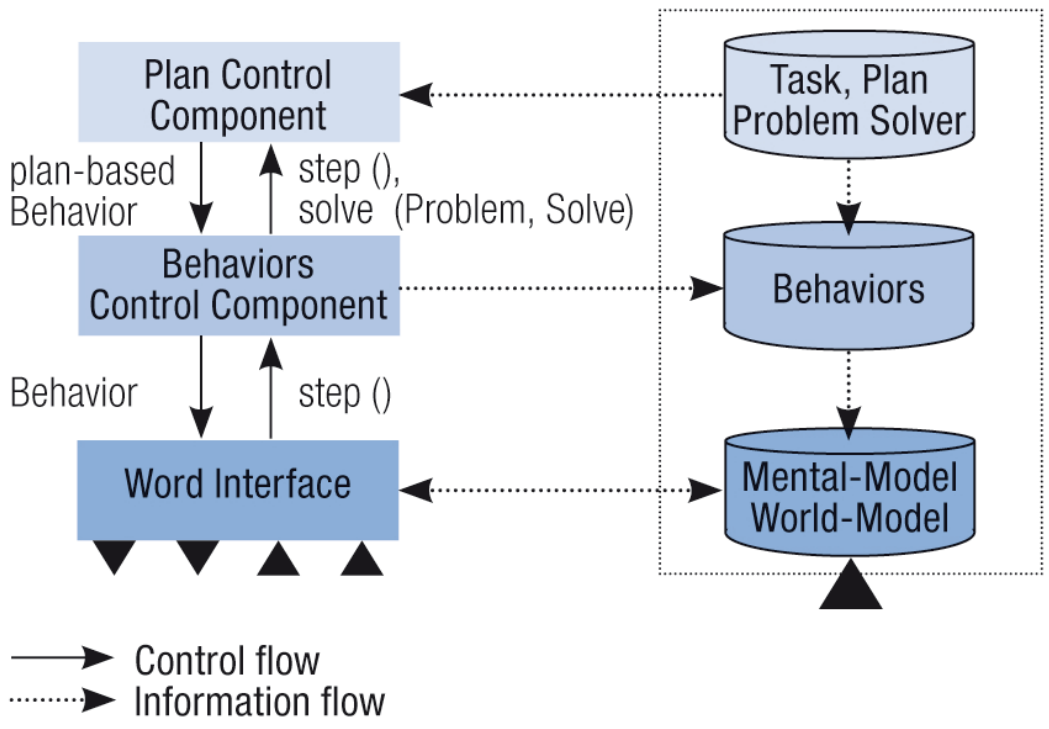


Figure 5.5: General structure of `Chu`'s Traffic Object Agent ([64] p. 80).

its goal on its level, then it can give the control to the upper level to solve the problem. For example, if the agent cannot change lanes through primitive behavior immediately on the BCC level, then the control flow is forwarded to the PCC level to activate plan-based behavior, which allows lane changing in the future.

In the WI component, the interface 'Action Executer' controls the simulation steps to perform the actions, the 'Message Interceptor' requests, replies and informs with the message type general-purpose computer programming language that is concurrent, class-based, object-oriented (JAVA) objects, and the 'Belief Updater' receives simulation messages during the pre-step to update its beliefs and during the post step to confirm that the agent action was executed successfully.

The BCC component receives a step message for activating a behavior from the WI. The control unit can choose and then activate a behavior which is stored in the knowledge base, depending on the agent's goal. The agent can have two types of goals: its individual and the joint goal. The individual goal of each vehicle is to drive its desired speed and the agent chooses its lane so that its utility is maximized according to its route preference. The general utility function and cost estimation function are described in the background 2.3.1.

An individual vehicle should always decide in which lane it should drive to reach and maintain its desired speed. A common way to do this is to use a utility-based decision strategy, described in the background 2.3.1. This means that the vehicle calculates the utility values of all lanes around it (left, right, and current lanes). A lane with maximal value is chosen (utility-based decision). The decision in such cases is based only on locally perceived information of the vehicle. Thus, it is not always optimal for the overall traffic state.

5.2.3 Patterns of Behavior

Patterns of behavior by `Müller and Pischel` [256] describe the procedural knowledge of an agent and can be activated by the agent itself in certain situations. Two basic behaviors of traffic participants are following and lane changing behavior, introduced in the background chapter 2.2.4 and 2.2.4. This thesis uses four basic behaviors for a vehicle agent (this was developed in collaboration with `Chu` [64]):

- *KeepOnCurrentLane* is mostly used by agents and is always executable (if the environment does not change). This behavior has no preconditions.It uses the model below for the calculation of the speed of the agent 5.7. This behavior does not require a lane change,unless the environment merges two lanes into one.
- *FollowVehicle* calculates the speed of an agent in case it needs to follow another agent. The difference to KeepOnCurrentLane behavior is that the following vehicle agent is not required to drive in the same lane as the leading agent. This behavior implements function $V^{ccr}_{V_a,V_c}(t+T) = max[V^{follow}_{V_a,V_c}, V_{min,V_a}(t+T)]$
- *DriveToTheLeft* and *DriveToTheRight* behaviors implement the lane changing model. Four mathematical conditions need to be fulfilled as a precondition. In contrast to traditional lane changing models, which have the problem that the gap for lane change is not big enough, a new cooperative method is proposed by `Chu` ([64], which is presented in the following.

The behaviors *DriveToTheLeft* and *DriveToTheRight* require coordination of several agents which implement the lane changing model. The first phase considers the motivation for the lane change and, in the second phase, the vehicle agent uses the Gap Acceptance Model (GAP) for measuring the size of the gap in the desired lane. If the gap is big enough, then the lane change is executed; if it is too small, the vehicle stays in its lane. Traditional lane changing models fail, because the gap for lane change is often not big enough, which is called the gap problem. Because no communication or coordination takes place, the vehicle drivers cannot negotiate about the necessary gap for a lane change. `Chu` ([64] p. 62) proposes a new cooperative method in which the autonomous vehicles use their coordination abilities to solve the gap problem.

This cooperative collision resolve (CCR) method allows the vehicles to cooperate to reach an adequate gap for a lane change, under the assumption that the vehicles were motivated to perform the lane change. Thus, lane changes become more secure and are executed efficiently. Mathematical conditions for a secure lane change need to be fulfilled. According to the model of Gipps [133], visualized in Figure 5.6, a vehicle V_c can change from lane 2 to lane 1 at a time t, if there exists a gap next to the actual position of the vehicle V_c.

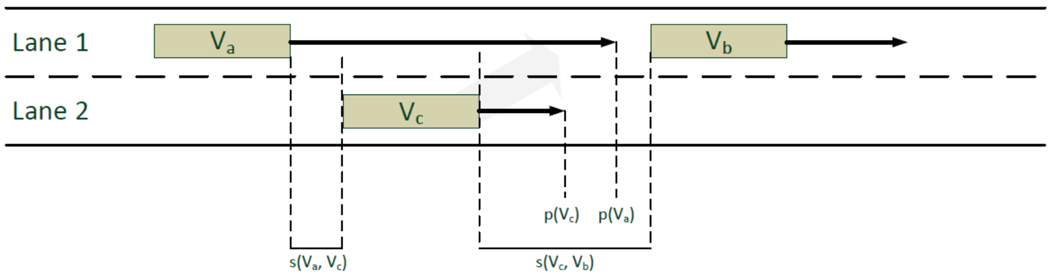


Figure 5.6: V_c tries to switch lanes from V_a to V_b from [140] p. 5

This gap can be expressed as follows:

$$s(V_a, V_c) = p_{V_c} - l_{V_c} - p_{V_a} \wedge s(V_a, V_c) \geq 0 \tag{5.1}$$

$$s(V_c, V_b) = p_{V_p} - l_{V_b} - p_{V_c} \wedge s(V_c, V_b) \geq 0 \tag{5.2}$$

where the parameters p_{V_a}, p_{V_c} are the positions of the vehicles V_a and V_c respectively. The term $s(V_a, V_c)$ is the distance between the vehicles V_a and V_c (compare Figure 5.6). The parameter l_{V_c} is the length of vehicle V_c.

The vehicle V_a can define vehicle V_c as its direct predecessor. In doing so, in case of a lane change, the vehicle V_c must assure an emergency braking distance, so that the vehicle V_a can also stop behind him. This is expressed in the following 5.3:

$$V_{min,V_a}(t + T) = max(V_{V_a}(t) + dcc_a \cdot T, 0) \tag{5.3}$$

The minimal speed to which the vehicle V_a can decelerate in time T is expressed in the following equation 5.4:

$$V^{new}_{V_a,V_c} \geq V_{min,V_a}(t + T) \tag{5.4}$$

where $V^{new}_{V_a,V_c}$ is calculated using the modified Gipps function 2.6, including communication and information exchange of their static characteristics

$$V^{new}_{V_a,V_c} = D_{V_a}T + \sqrt{D_{V_a}T - D_{V_a}\begin{pmatrix} 2\left(X^{new}_{V_b}(t) - X_{V_a}(t) - L_{V_b} - M_{V_a}\right) \\ - V_{V_a}(t)T - V^2_{V_b}(t)0.5 \end{pmatrix}} \tag{5.5}$$

The condition 5.4 does not need to be fulfilled if the vehicle V_c has no successor V_a on its final lane.

The vehicle V_c can define vehicle V_b as his next predecessor. The condition for vehicle V_c is analog to the condition for vehicle V_a:

$$V_{min,V_c}(t+T) = max(V_{V_c}(t) + dcc_c \cdot T, 0)$$
$$V^{new}_{V_c,V_b} \geq V_{min,V_c}(t+T)$$

whereas $V_{min,V_c}(t+T)$ is the minimal speed to which the vehicle V_c can decelerate in time T. The speed $V^{new}_{V_c,V_b}$ is calculated by the function 5.5. The condition 5.6 does not need to be fulfilled if the vehicle V_c has no successor V_b on its final lane.

Which corresponding behavior needs to be activated depends on the actual state of an agent. A behavior is defined so as to be executed in a certain situation. The interface of each behavior mentioned consists of the following functions (cf. [64] p. 85f.):

1. `checkprecondition():` the function checkprecondition checks whether the actual environment condition and the mental state of the agent make it possible to execute the behavior.
2. `getFailure():` in case the function checkprecondition discovers an error, the execution of the behavior is prevented and the failure can be identified through this function getFailure.
3. `preExecute():` this function defines all actions which the agent needs to execute before its behavior is performed.
4. `execute():` this function executes the behavior. In case an agent is allowed to execute the function of a behavior, then the function getFailure returns the value null.
5. `postExecute():` this function defines all actions which an agent needs to conduct after its behavior was performed.
6. `getUtility():` this function defines the utility value of a behavior which the agent receives after it executed the behavior. The utility is only calculated if the behavior is executable.
7. `getResult():` this function returns the result of a behavior. All behavioral results have the following form:
   Result:

speed $\rightarrow n$	; n is a positive floating-point number
direction $\rightarrow i$	; i is a number of the quantity $[-1, 0, 1]$
	; whereas the numbers $-1, 0, 1$ stand for
	; the left, straight and right direction

A lane change decision can be done with the *c*ooperative *c*ollision *r*esolve method. It creates a joint plan for all vehicles in a group with static predefined actions for each vehicle, but due to dynamic changes it is not efficient or even executable. The following utility function is based on a utility-based lane change decision:

$$Util_x(V) = [V^{ccr}_{V,V_{pre}}(t+T) - V^{ccr}_{max,V}(t+T)] + [V^{ccr}_{V_{succ},V}(t+T) - V^{new}_{max,V_{succ}}(t+T)] \tag{5.6}$$

where $Util_x$ is the utility value of the lane x. $V^{new}_{max,V_{succ}}(t+T)$ is the maximal speed which the vehicle V can reach in the time period $t+T$. This depends on the acceleration behavior of the autonomous vehicle, which can be expressed in the following function (modified from the original `Gipps` [132, 133] function without the parameters 2.5 and 0.025 for imitating human behavior):

$$V^{new}_{max,V_a}(t+T) = min(V_{V_a}(t) + acc_a \cdot T, ds_a) \qquad (5.7)$$

This formula calculates the maximal reachable speed of vehicle V_a in the time period $(t, t+T)$. The internal minimizing function $V_{V_a}(t) + acc_a \cdot T, ds_a$ computes the maximal speed which a vehicle V_a can accelerate to in the time period $(t, t+T)$. The parameter ds_a is the desired speed of the vehicle V_a. The function 5.7 correlates the maximal speed $V^{new}_{max,V_a}(t+T)$ with the minimum of reachable speed $V_{V_a}(t) + acc_a \cdot T$ and the desired speed ds_a of the vehicle V_a.

In the utility function, 5.6 $V^{ccr}_{V,V_{pre}}(t+T)$ stands for the potential speed which the vehicle V can reach if it changes to lane x. $V^{ccr}_{V_{succ},V}(t+T)$ is the potential speed of the successor V_{succ} of the vehicle V on lane x.

Compared to the structure of `Müller and Pischel` [256], no abortion criteria for stopping a behavior is implemented in `Chu` ([64] p. 86), because the behaviors are only short-termed and executable within a simulation step. A behavior terminates if the preconditions are not fulfilled. For example, a plan is pre-planned with three sequential behaviors FollowVehicle for three seconds, but due to the ATSim technology, the environment information can only be updated after one second. Assuming that after the second execution of FollowVehicle, the actual information shows that the leading vehicle changed lanes, then the next planned behavior terminates because the precondition that the leading vehicle is in the same lane is not fulfilled.

5.3 Group Planning

The first process **group context** of the group life cycle illustrated in Figure 5.3 includes aspects of the outer surroundings, the (simulation) environment, or the larger organization, which have an influence on group performance, operation and effectiveness. The following elements of the group context influence the group in its planning phase. They encompass the physical environment described in Section 5.3.1 that fits the group's needs, a supportive organizational traffic cooperation as in Section 5.3.2, information as in Section 5.3.3, including feedback about performance, a clear mission and shared values, and rewards and recognition as in Section 5.3.4 which are consistent with the group objectives and design.

For group planning, as a member of a group, a vehicle receives global coordination of leaders via group lanes to help it avoid conflict situations. However, it is not an obligation of the vehicle to obey the coordination of its leader. In some cases a vehicle should consider to choose its future lane based on its own utility rather than following the coordination of the leader. In this matter, the question is raised (own publications [64, 140]): "When should a vehicle follow the coordination of the leader and when not?" In order to answer this question a state-based lane choosing strategy was developed. A vehicle is always in one of the four states `Grouping, Forming, Overtaking, or Free Driving`. The coordination strategy or the lane-utility decision presented in Equation 5.6 are the base for each state in which the vehicle chooses lanes.

- *Grouping:* The vehicle is not grouped yet. This state allows the vehicle to drive in the lane which permits it to reach its desired speed (utility-based decision).
- *Forming:* The vehicles belongs to a group, but does not drive in the group lane. The vehicle follows the coordination of the leader and ignores its own utility in this state. Thus, if required the vehicle reduces its current speed and accepts to change to the group lanes.
- *Overtaking:* The vehicle belongs to a group and is driving in one of the group lanes. A utility-based decision can be made by the vehicle only on its group lanes. If the utility can be maximized, the vehicle will try to change to another lane.
- *Free Driving:* The vehicle is a group member and is driving in front of all the others. Therefore the vehicle can ignore the coordination of the group leader who is driving behind and uses utility-based decision making for choosing its future lane.

The group does not control the group context, but it might influence the outer surroundings, the (simulation) environment, or the larger organization to create a more supportive one. Understanding the group context helps to identify where the influencing aspects help or hinder the group's efforts to improve effectiveness.

The other way around, elements of the group context can contribute to group problems or support increased effectiveness, but the group context contains elements that influence but do not control the group.

5.3.1 Physical Environment

Depending on real traffic conditions, traffic with few vehicles can rely on static traffic control, whereas with many vehicles dynamic and active traffic management with the help of new technologies is required. With too many vehicles, dynamic traffic management can only protect from degradation (Network Fundamental Diagram [162]). Four traffic conditions and their according actions can be extracted (which also correlate with the Levels of Service presented in the background section 2.2.3):

1. undersaturated - maximize the speed
2. saturated - maximize the capacity and throughput
3. over-saturated - line-up management and metering

4. blocked - call for assistance i.e., police or change medium i.e., use public transport

This thesis assumes many vehicles and saturated or over-saturated conditions. Thus, the environment input is fixed, but could be adapted for changes in future work.

The environment can be a particular area of the real world or a simulation of a scenario. The latter is used in this thesis with traffic simulators based on real world data from Hanover, Germany. The traffic environment provides the technical and physical infrastructure in which the traffic participants operate. Traffic participants, especially vehicles for this thesis, can be represented by agents as described before in Section 5.2. Agents use the environment to achieve their individual or collective goals, which are usually related to objects of the surroundings with various opportunities in the physical or virtual place. The environment does not require a unified objective or architecture and does not provide a coordinate system for agents to participate in. It has the scope of defining a boundary, so agents may enter, leave, and return later to the environment scenario, which is a part of the whole environment. Scenarios are extracts from the environment and comprise a system of systems (defined in Section 6.1). The environment consists of a structured collection of scenarios describing predefined infrastructure elements for particular coordinated activities or for achieving certain subgoals. In many cooperative applications the agents need to adapt to environmental changes.

Secondly, the environment is structured with organizations like the postal service (and, in future, may provide institutions such as a court of justice) to foster coordination and regulate beyond physical and technical constraints the interaction of autonomous, heterogeneous agents. Organizations regulate the agents' behavior by functional (global goals, plans, missions, schemes, preferences), structural (groups, links, roles, compatibilities, multiplicities, inheritance) and deontic (permissions, obligations- but agents' autonomy needs to be considered!) specifications (cf. [45] p. 16). The goal of the environment with its participants is to balance different interests and to establish and sustain certain notions of stability. Within the environment organizations structure the grouping and collaboration of agents. Organizations are introduced in Section 5.3.2. MAS Services provide an organization, interaction and environment infrastructure.

Traffic control depends on the input of hardware technology which perceives the environment, such as sensors, communication and computation as the actuators, and uses software methodology with intelligence, data processing and control strategy as the output reaction to the environment. Basically, the real world is measured with the help of a computer where the input, the process and the output of an automatic control system are calculated. Then, after data processing and control strategies depending on the traffic control goals, the output is returned to the traffic environment.

Cooperative Traffic

Increasingly large information environments such as intelligent traffic and the opportunities of MAS are a change for manageable distributed control in traffic. Planned traffic environments are too large, complex, dynamic and open to be managed centrally or with predefined techniques ([373] p. 112). Entities of embedded intelligence are well-suited, though, to autonomous agents that find, transmit or manage information. Because of the nature of the environment, agents must be long-lived, adaptive and social in order to interact and coordinate global and individual goals.

In order to describe traffic management as a Multi-Agent System, it includes the model of a system of traffic participants in terms of a network of rational agents.

1. Classic control and regulations need to be translated into institutional norms of traffic agent behavior, with traffic rules and resulting valid, legal and meaningful actions for the different types of environments and topologies. A traffic simulation system could provide general control and regulations with preset objects i.e., traffic lights, street segments with lanes, priority streets and actions.

2. The cooperative intelligent behavior of agents with belief representation (knowledge of the single agent about the system - information model), cooperation (for example, decentralized grouping) and intelligent cooperative decision making (for example, routing) and learning (adaptive search for the best policy) needs to be added to the traffic simulation system as a multi-agent environment.

3. Information exchange between agents including rules of knowledge exchange (when and what to send to whom) and communication protocols of the knowledge exchange (compact and clear to others) need to be provided for simulation. The facilitation of joining beliefs (how to deal with received information) would also be desirable, which could also be handled in the agent architecture.

The innovative approach is that agents join a group autonomously and form a small society which defines the social context in which the agents interact. Social commitments are commitments of one agent to another agent, studied in `Jennings` [191], and of an individual to groups, studied in `Castelfranchi` [59]. A cooperative group is based on a common goal and mutual dependence ([59] p. 45). Social commitments constrain the behavior of agents depending on the society goal. Cooperation is a form of mutual dependence. Beyond social dependencies, social laws may govern the behavior of agents in a society.

Modeling Urban Traffic

The traffic system can be seen as an interconnected, geographically distributed system where the traffic participants (here: vehicles) are capable of processing local real-time information independently and of interacting in a cooperative manner.

The **world definition** is constituted of MAS and the environment:

$world = (environment, mas)$

The environment is constituted of objects:

$environment : E = (object_0, object_1, ..., object_n)$

where an object has a name, some states, properties and relations with other objects. In case of an activated property, it can modify the states or activate other properties of this object or of its relation. An object has no goal.

There is a network of routes which can be depicted as a graph. As there can be several lanes on a route, these routes are described by an identifier:

$$f(x, y) = \alpha_1 \frac{ds_x - ds_y}{ds_{desiredspeed}} \tag{5.8}$$

Vehicles follow their individual route on streets or roads[1] and their lanes on the graph. Routes can be seen as sequences of nodes of a graph.

Definition 5.1 (Route) *route*

Furthermore, they have the preference to pass through each of these segments at their desired speed, which is defined by a natural number. It can be considered as the remaining time to reach the next node and thus to enter a new section. The maximal speed is defined in the same way, i.e., the quickest possibility for the vehicle to reach the next node:

Definition 5.2 (Vehicle) *Let $G = (N, E)$ be a graph with multiple nodes and edges. Then $RP = \{\langle (s_1, sp_1^{des}, sp_1^{max}), (s_2, sp_2^{des}, sp_2^{max}), ..., (s_n, sp_n^{des}, sp_n^{max})\rangle | (\forall_{1\leq i\leq n} sp_i^{max}, sp_i^{des} \in \mathbb{N}) \wedge (\forall_{1\leq i\leq n-1} \exists (s_i, s_{i+1}, id) \in E)\}$ denotes the set of route preferences. $V \subseteq \{(v, sz, rp, (s_0, s_1, l), sp^{act}) | v, sz \in \mathbb{N}, sp^{act} \in \mathbb{N}^\infty, (s_0, s_1, l) \in E, rp \in RP\}$ is the set of vehicles with the individual vehicle having identifier v, size sz, and route preference rp. It is currently driving at speed sp^{act} (actual time to reach the next node) in the section between s_0 and s_1 on lane l.*

Besides the route network and the vehicles, a traditional traffic system also has controlling elements like traffic lights or signs. Here, it is assumed that such signs are always located at nodes, influencing the traffic before or after the node.

[1]Streets or Roads? In theory a road is different from a street, although both terms are often applied to the same thing. Looking at definitions from city planners the difference is a matter of place and purpose.

The word street is specifically applied to urban lanes. Streets interlink people for interaction, whereas roads are the highways which connect towns and cities for travel.

Due to urbanization, roads can serve the purposes of streets but do not have their names changed.

Roads connect two distant points - two cities such as Hanover and Berlin, for example. Each of those cities has streets: paved roads surrounded by houses and other buildings. Historically, the word street used to mean the pavement and the buildings that made it into a street. However, today many paved roads exist with buildings on them because cities grew and suburbs were merged.

Definition 5.3 (Traffic System) *Let $G = (N, E)$ be a graph with multiple nodes and edges and V be a set of vehicles. Furthermore, let C be a set of controlling elements. Then $TS = ((N, E), V, C, contr, cap)$ is a traffic system with $contr : N \to \mathbb{C}$ assigning controlling elements to nodes and $cap : E \to \mathbb{N}$ defining a capacity for each section of the graph.*

Decentralized Optimization

A road network without traffic rules is considered in a receding time horizon made of T time periods. Let a triple $G = (V, E, M)$ be a connected digraph representing the road network where $V = \{v_1, \ldots, v_m\}$ is the set of nodes representing intersections, with m the number of nodes, E the set of directed arcs (edges) $(i, j),\ i, j, \in V,\ i \neq j$ representing roads connecting intersection i with j, and M is the adjacency matrix with cost being its element $M[i][j] \geq 0$, $i, j = 1, 2, \ldots, m$ in which f_{ij} denotes the cost (e.g. distance or travel time) of arc (V_i, V_j).

$$M[i][j] = \begin{cases} f_{ij} & (V_i, V_j) \in E \\ \infty & (V_i, V_j) \notin E \end{cases} \tag{5.9}$$

The cost f_{ij} can be an average travel time function $A_{ij}(x_{ij}(t))$ of an edge (i, j) which each user of arc (i, j) experiences when $x_{ij}(t)$ units of vehicles flow along the arc. $A_{ij}(x_{ij}(t))$ is, in general, an increasing nonlinear function because of the effects of congestion on travel time for each user of any arc. The form of the cost function is used by the U.S. Federal Highway Administration traffic assignment models:

$$f_{ij}(x_{ij}(t)) = \int_0^{x_{ij}(t)} A_{ij}(t)dt = a_{ij}x_{ij}(t) + \big(b_{ij}/5\big)(x_{ij}(t))^5 . \tag{5.10}$$

a_{ij} and b_{ij} are empirically determined parameters for each arc which are computed from its length, speed limit and geometric design including number of lanes and traffic lights.

The cost function represented in this way has an upper limit given by the maximal capacity of each arc. Let $C = (c_{ij})$ denote the capacity vector, where $c_{ij} \geq 0$ denotes the capacity of arc $(i, j) \in E$. For simplicity and without loss of generality, assume that nodes $1, \ldots, p,\ p \leq m$ make a set of origins O and destinations S. Let $x^s_{ij}(t)$ be the flow along arc (i, j) with destination s at the beginning of time interval t, and, similarly, let $x_{ij}(t)$ be the total flow along arc (i, j) at the beginning of time interval t such that

$$x_{ij}(t) = \sum_{s=1}^{p} x^s_{ij}(t),\ \forall (i, j) \in E. \tag{5.11}$$

Let $D(t)$ be a $p \times p$ matrix whose $(i, j) = d_w$ entry indicates the number of units which must flow between nodes i (origin O) and j (destination D).

Each individual request within d_w has a time window $\omega_a = [\underline{\omega_a}, \bar{\omega_a}]$, where $\underline{\omega_i}$ is the earliest time to begin the trip and $\bar{\omega_i}$ the latest time to arrive at the destination, which has to be larger than or equal to the expected time of arrival at the destination. These time windows are 'soft constraints', as a vehicle can arrive later than $\bar{\omega_i}$. Origin-destination flow matrix $D(t)$ is grouped into a number of (possibly overlapping) time periods (e.g., all vehicles with the same time window).

Then the capacity constraints for each arc (i, j) in the network are

$$c_{ij} \geq \sum_s x^s_{ij}(t),\ \forall (i,j) \in E. \tag{5.12}$$

For each $w \in W$, let P_w denote the set of available paths joining $O - D$ pair w. Let $r = \sum_{w \in W} |P_w|$. For a given path $k \in P_w$, let $I_k = \left(e^k_q, \ldots, e^k_x\right)$ be (O-D) the itinerary assigned based on the actual traffic situation in the network.

Let $h^{ij}_k(t)$ denote the traffic flow on edge (i, j) caused by path k at the beginning of time period t.

All the paths can be gathered in the vector $h(t) = (h_1(t), \ldots, h_m(t)) \in R^m$ called a path flow, similar to `Tian and Xu` [343]. The path flow vector h induces an arc flow $x_{ij}(t)$ on each arc $e = (i, j) \in E$ given by

$$x_{ij}(t) = \sum_{w \in W} \sum_{k \in P_w} h^{ij}_k(t). \tag{5.13}$$

If a demand of network flow is fixed for each $O - D$ pair w, the path flow h satisfies demand constraints $\sum_{k \in P_w} h_k = d_w,\ \forall w \in W$ and capacity constraints. That is called a feasible path flow.

Let $A(i) = \{k \in A \mid k$ points out of node $i\}$ and $B(i) = \{j \in A \mid j$ points into node $i\}$. $d_{B(i),i}(t)$ is the decision variable for the number of vehicles that are admitted to nodes $k \in B(i)$ to i onto the arc ij during the time period t.

If there were no capacity constraints on arc (i, j), then the unconstrained decision variable d^*_{ij} would be

$$d^*_{i,j}(t) = d_{B(i),i}(t) = \sum_s \sum_{k \in B(i)} x^s_{ki}(t-1),\ \forall i \in V \tag{5.14}$$

However, due to the capacity constraints of arc (i, j), c_{ij}, the actual vehicle flow on arc (i, j), $x_{ij}(t)$ is lower than or equal to the arc capacity, i.e., $x_{i,j} \leq c_{ij}$, while the decision variable for the number of vehicles admitted to arc (i, j), $d_{i,j}(t)$, is equal to or, in general, lower than the unconstrained decision variable $d^*_{i,j}(t)$. Hence, a negotiation process must be implemented between vehicles and infrastructure to arrive at a set of local solutions that together satisfy the constraints in (5.12).

The flow constraint on arc (i, j) from one time period to another is:

$$\sum_{s=1}^{p} x_{ij}^{s}(t+1) = \sum_{s=1}^{p} x_{ij}^{s}(t) - g_{ij}(t) + \sum_{k\in B(q)} d_{ki}(t) + \\ + \sum_{s=1}^{p} D_{i,j}^{s}(t),\ \forall\ (i,j) \in E \setminus s \in S,\ t = 1,\ldots,T-1 \tag{5.15}$$

where $g_{ij}(t)$ is an exit function of arc (i,j) at the end of time period t which is influenced by the actual flow at arc (i,j), but also by the vehicle flow at the node j, $x_{ij}(t)$ and all the connecting arcs $A(j)$ going out of arc (i,j). Traffic propagates over every intersection from incoming to the forward outgoing sections. Decision variable $d_{ij}(t+1)$ depends on the exit function $g_{ij}(t)$, and the exit function $g_{ij}(t)$ depends on the entry function of $d_{jk}(t-1)$.

$$g_{ij} = \min\left\{\frac{f_{ij}^{x_{ij}}(t)}{s_l(v_e)}, d_{j,A(j)}(t+1)\right\},\ \forall (i,j) \in E \setminus s \in S \tag{5.16}$$

In the above formula, $s_l(v_e)$ is a sum of the average vehicle length and the distance between vehicles. The latter is a function of the momentary average arc velocity $v_e(t)$ but for simplicity a constant value is assumed.

For the optimization of the throughput of the network, one possible parameter could be utilization. It could be defined as the ratio of the difference between the number of vehicles arriving at the arc and the ones leaving it and the maximal number of vehicles allowed in the arc, i.e., its capacity.

$$\gamma_{ij}(t) = \frac{d_{ij}(t) + x_{ij}(t-1) - f_{ij}(t)}{c_{ij}} \tag{5.17}$$

It is assumed that the utilization of an arc is low, that is, that there is no traffic congestion at that part of the network.

The goal is to achieve a vehicle flow of minimum cost such that each vehicle follows one route from its origin and terminates at the destination position with the feasibility of the time schedule and constraints on vehicle flow being satisfied.

The T-period multiple origin/multiple destination problem denoted by P is minimizing the overall cost (average arrival time) of origin-destination (O-D) pairs.

This process is supervised by local decision makers that assign vehicles to the arcs (i,j) for each time period on the basis of their requests, guaranteeing the fulfillment of the constraint on the limited infrastructure capacity and flow. In the next section, a model for such negotiation is provided.

The static objective function of the road network (disregarding the temporal changes of the traffic) can be expressed as: (P_i) :

$$\begin{aligned} \min f(x) &= \min \sum_{(i,j)\in A} f_{ij}\big(\sum_{s=1}^{p} x_{ij}^{s}\big), \\ \min f(x) &= \min \sum_{(i,j)\in A} [a_{ij}\big(\sum_{s=1}^{p} x_{ij}^{s}\big) + (b_{ij}/5)\big(\sum_{s=1}^{p} x_{ij}^{s}\big)^{5}] \end{aligned} \tag{5.18}$$

subject to:

$$D^s(i,j) + \sum_i x_{ij}^s = \sum_k x_{jk}^s \ , j = 1, \ldots, n, \ s = 1, \ldots, p, \ j \neq s \tag{5.19}$$

$$x_{ij}^s \geq 0, \ (i,j) \in A; \ s = 1, \ldots, p. \tag{5.20}$$

Optimizing Vehicle Agents

The same principle of two level optimization holds. On the upper level, the network decides on the flows which are supposed to pass the O-D pairs, and, at the lower level, individual vehicles decide on the individual sequence in passing these flows based on a social benefit function that guarantees equality.

Let $a_i(t)$ be the collaborative vehicle agents present in the road network at time t. Furthermore, the system is open, so the number of vehicles in the system is variable in time.

Each agent $a \in A$ is described by the tuple

$$a = \{p_a(1), p_a(t), \ p_a^{\theta}, \ w_{max}^{[a]}, \omega_a, trav_a(t), l_a\} \ , \tag{5.21}$$

where $p_a(1)$ is the origin position, $p_a(t) \in S$ being the position of vehicle a at time $t = 1, \ldots, T$, with $w_{max}^{[a]}$ its maximum movement distance (maximum step size) in each assignment interval. $p_a^{\theta} \in G$ is a destination position θ of agent a and $trav_a$ is actual time (or distance, depending on individual objective function) traveled until time period t. We assume that, at time $t = 1$, all vehicles are positioned at their origin positions $p_a(1)$. At any time t, each agent a knows its origin position $p_a(1)$, momentary position $p_a(t)$ and its destination position p_{θ_a}. Note that the number of locations θ is identical to the number of vehicle agents a in the system. We assume that all vehicles have the same average length l_a. We also assume that vehicle agents are collaborative and, if needed, unconditionally share available information with their neighboring vehicles and infrastructure.

Let $dist_{a\theta}(t)$ be the shortest distance between the momentary $p_a(t)$ and the destination θ_a position. There is the issue of jammed sections- search to avoid the jammed areas using the second shortest route.

Each vehicle searches for the lowest cost path h_k where $k \in P_w$. These individual preferences are compared with the capacity limits of the network. Then they are given priorities by the network, respecting a social benefit function.

The solution cost typically relates to the total travel time, total distance traveled, and lateness at destination. New vehicles continuously enter the system over time and must be directed to their destinations in real time.

The problem of dynamic assignment of resources is considered for a set of vehicle agents, which in this case are a set Φ of semaphore light locations and road sections and available times to pass these.

Vehicle agents optimize their route based on the constraints of sequentiality, i.e., the next section in the itinerary cannot be passed if the section before has not been passed in the period before, so there is temporal and spatial sequentiality in each itinerary.

The question here is, should the network produce these paths for every time period and update them dynamically so that the network offers the service of passage of the network or, should the vehicles individually calculate their routes as they come.

Each vehicle's objective is to minimize the cost of travel which can be travel time, travel distance, consumed energy or other things. These objectives can be combined into an individual, multiple objective goal function.

Agents are collaborative and only receive information through their local interaction with the connected agents in the environment. T is the upper estimated time boundary in which all the agents present in the system could reach their assigned target locations.

Classical shortest path algorithms can be used to calculate the best route between the current position and the destination in a network. There are several known algorithms for routing. Since Dijkstra is a de facto standard in this field, this algorithm is used for the purpose of this thesis.

Coordination through Communication

Traffic participants' interactions are mostly local in nature. Thus, this leads to the setting where most of the participants usually have intense interaction within some range based on their geographical position. In contrast, outside this range almost no interaction takes place. In this context, distributing the computation among vehicles and balancing their communication load can increase the overall throughput, robustness, and flexibility of the traffic system. In such a distributed setting, centralized coordination is obsolete, becoming more dynamic and decentralized instead.

For the aforementioned reasons, this thesis proposes a dynamic and distributed traffic system with vehicle group models applicable to all traffic participants. The key to the group life cycle model lies in the distribution of the traffic-related decisions to allow as high autonomy as possible regarding local decisions.

Grouping extends the functionality of routing for speed adaption and lane changes as shown in my own previous work ([140] and [141]). In `Görmer and Müller` [140], more detailed group methods are described for group-oriented traffic coordination and its architecture. Depending on where the vehicles are at which time and their constraints, vehicles form one group for a common goal.

The goal of groups is to cross several intersections using the green wave and avoiding stop times by selecting the proper speed.

For group formation, the additional necessary information about the signal plans of the traffic lights is sent by Road Side Units (RSU). For agent based models, the requirement is perfect communication between the single agents, here, the vehicles. Group formation with realistic communication was tested in urban traffic. To avoid overloading the communication channel, messages are not sent as often as desired by the agents. The data format for the communication is a simplified, predefined message format - modified CAMs.

5.3.2 Traffic Cooperation

Group culture (cf. [306] p. 20) is defined as a fundamental set of values and beliefs which are shared by group members and which guides their behavior. A value is an assumption about what is worthwhile or desirable and a belief is an assumption about what is true. The group is effective if every member knows the core values and beliefs and acts based on them.

Coordination with organizations and institutions with norms are important governance elements. This section focuses on the first elements of coordination with organizations for the design of cooperative vehicle groups and the use of open systems, in this case the traffic system. The latter, institutions with norms, can be tackled in other or future research with the idea of using a three-fold separation of powers: executive (i.e., the police for direct law enforcement and punishment), constitutional (i.e., creating laws based on conventions and group behavior) and legal (i.e., a judge decides in individual conflicts) priorities modeled with the agent paradigm.

Cooperation takes place in division of tasks and requires coordination according to `Laux` (cf. [223] p.) and `Kieser` (cf. [203] p. 100f). There are interdependencies between each sub-task which fulfill an overall goal. Coordination mechanisms and instruments are described as rules which serve the overall goal for managing the control of interdependencies and the negotiation and orientation of the tasks.

Organization is seen as a system of decisions (cf. [223] p. 13). According to `Jost` (cf. [196] p. 62f), a differentiation can be made between two decision making systems:

- decentralized decision making, based on making decisions independently and, therefore, making autonomous decisions
- centralized decision making, when a superior entity makes the decision and gives orders in a hierarchical coordination.

The theory of agency ([203] p. 50) is a contract theory view of an organization where the employer (also known as the principal) hires a contractor (agent) who performs in the interest of the principal, which is fixed in a contract i.e., a work contract. Each party maximizes its utility while deciding on and fulfilling the contract. Different interests and asymmetric information (the agent having more information), such that the principal cannot directly ensure that the agent is always acting in the principals best interests, can cause a conflict. The theory of agency is seen as a formal structure of an organization which regulates the activities of its members. This theory is not efficient for complex and unstructured tasks and needs a stimulus to affect the behavior of the setting of goals by the principal or the action of the agent.

Next-generation traffic management systems will also incorporate decentralized aspects such as the on-board intelligence and communication capabilities of vehicles and traffic infrastructure (cf.[140]). In this thesis, a multi-agent approach is investigated, allowing vehicle agents to form groups in order to coordinate their speed and lane choices. Each autonomous vehicle agent has its individual behavior, described in 2.3.1. The hypothesis is that a decentralized approach based on a cooperative driving method will lead to a improved traffic flow which causes higher speeds and less delays. The focus on automated vehicle decision models contributes to a group-oriented driving method with vehicle agents that perceive their environment 2.3.2 and exchange information 5.3.1.

5.3.3 Information System

This thesis discusses the aspect of decentralization and the resulting necessity of information management within and between the decentralized elements.

Information includes many things, including the belief about facts, the viewpoint and the reasoning. Assumptions about information can be shared to others (refer to [306] p. 43):

- I have some relevant information; others also have relevant information
- Each individual may see things that others do not
- Differences are opportunities for learning
- Participants are trying to act with integrity given their situations

Organizations and group behavior facilitate information sharing and problem solving. Basic group behaviors are set for mobile, situated, spatially interacting and embodied agents. For the purpose of classifying group behavior, my research is aimed at finding common properties across various domains (namely AI, Traffic, Economics and Psychology) of multi-agent interaction.

Real-time information exchange and processing, as well as communication with the traffic management center (TMC) and other vehicles is supported by on-board units (OBU), integrated in each vehicle, and Road Side Units (RSU, which can be combined with traffic lights) in the street network for traffic strategies and communication infrastructure. The OBU allows individual vehicles to receive communication data and process it to make autonomous decisions about strategic grouping and then propose them as recommendations to drivers or, in future scenarios, perform autonomous driving.

From the global perspective, the TMC receives information from Floating Car Data (FCD) of the vehicles, processes it and makes global optimization decisions, which are returned back to the vehicles in the form of messages and signals from the traffic control infrastructure.

The focus is on the vehicles' autonomous group formation, done with communication. Vehicle groups are beneficial for individuals for coordinating their speed and for traffic management to make corresponding recommendations to drivers. Communication between vehicles is based on V2I and V2V protocols, which connects the TMC with the vehicles.

5.3.4 Rewards and Recognition

Rewards and recognition are based on strategic productivity and quality objectives and ensure a positive, productive and innovative organizational climate. This is especially true when persons are involved, whereas for automated systems the rewards and recognition need to be consistent with the objectives and design utility function, joint utility function and general global optimization. The reward structure can reduce the level of environmental stress and improve group task characteristics. The program needs to feature a healthy blend of both individual and group recognition and a mixture of rewards, i.e., monetary and non-monetary. Rewards are often connected to learning environments.

For the successful design of multi-agent learning algorithms, analyzing the effectiveness of agent reward structures is crucial (cf. [3]). The reward properties that lead to good system behavior need to be analyzed, i.e., properties promoting coordination among the agents. This happens if an agent takes an action which increases the value of its reward and the overall system reward also increases. The second property promotes rewards which are impacted by the agent's own actions. The reward properties describe the trade-off between the level of coordination among the agents and the difficulty of the learning problem each agent faces. Furthermore, the learning is a function of the problem domain and the agents' reward structure. Thus, the reward is independent of the learning algorithm. This is particularly helpful in continuous, dynamic, stochastic domains.

In general for large systems, the full system reward has little impact on the full system, but individual actions have a big impact on agent behavior. The rewards and recognition balance service and productivity and provide a learning opportunity.

Rewards and recognition are an instrument for meta-coordination and usually designed as part of the agent's action choice: the reward effect of decision making - also for group forming or the quality of groups - leads to adaptive (group) methods, as illustrated in Figure 5.9. Note that this is supporting group operation and needs to be validated. This is subject to future work.

5.4 Group Methods and Structure

Group methods and structure is the second factor which contributes to group effectiveness (subject to simulation and not evaluated in this thesis). The structure refers to the relatively stable characteristics of a group: mission and vision, task, membership, roles, available time, shared values and beliefs, and norms.

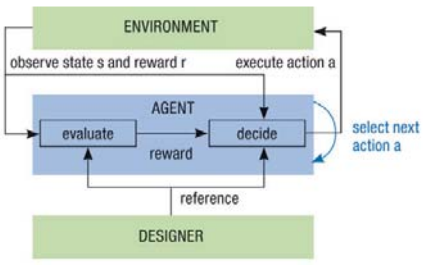


Figure 5.7: Single Agent Decision Loop.

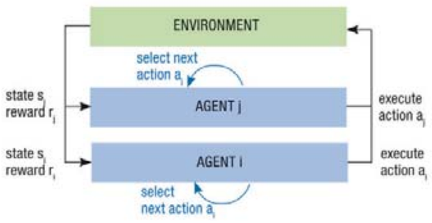


Figure 5.8: Multi-Agent Decision Loop.

Figure 5.9: Rewards of Single Agents and Multi-Agents for their Decisions.

An important key is to understand the dynamic group relationships which create the structure, because changing the relationships during the activity changes the structure.

Dynamic organizational models, like dynamic grouping and group coordination, can be modeled using Multi-Agent-Systems in interconnected systems like traffic and logistics. Decentralized concepts will highly influence future mobility and logistic supply chains.

Interaction strategies are the link between the macro and the micro view, i.e., efficient, goal-oriented communication poses challenges in gathering and processing big amounts of data. Group-based coordination and cooperation are important factors in those systems. For example, through effective inter-group communication between group leaders, instead of many to many $n : m$, platoons in front of a traffic light can be more efficient thanks to less communication complexity. Thus, research of protocols, methods, concepts, and algorithms for dynamic grouping and group coordination is investigated, presented and evaluated. One application field is in decentralized urban traffic management, to coordinate vehicles in platoons in order to optimize traffic flows, which I do with group formation/coordination methods like swarm algorithms.

Group work has become a widely accepted metaphor for describing the nature of multi-agent cooperation. The notion of group work involves shared knowledge, goals, which are communicated, resulting in activities that function as the glue that binds group members together. By virtue of a largely reusable explicit formal model of shared intentions, group members coherently attempt to manage general responsibilities and commitments to each other. When unanticipated problems arise both enhance performance and facilitate recovery. Group work for software agents alongside people requires the ability to carry out complex real-world tasks. Participating software agents must act naturally in such systems. Thus, tools and methodologies assure reliability and safety in working together including independent design of agents.

In this context, the focus of the work was to introduce an agent-based organizational model for dynamic traffic. The resulting dynamic group formation mechanism relies on decision support. Thus, the main line of research is the design of a fully decentralized and distributed approach, which formalizes a scheme for an open environment combined with multi-agent interaction. Moreover, in open systems, due to the transient nature of organizations, groups must be formed and able to cope with changing environmental conditions during run time. Therefore, in the following, a solution concept for dynamic traffic management is proposed that stems from the existent stability approaches in coalitional games. The solution, deployed in distributed environments, introduces negotiating agents with a corresponding algorithm. This algorithm functions as an open organizational adaptation within stochastic scenarios to meet the desired functionalities in order to achieve stable configurations. Empirical results are provided for validating the approach. They show significant improvement in organizational efficiency for the complex domain of dynamic traffic systems.

Different methods for group work (i.e., role-based vs. goal-based) and group formation algorithms are presented. The goal is creating a framework for dynamic group formation which uses both role-based and goal-based methods.

Lots of systems intend to coordinate participating agents for solving tasks either more efficiently with the help of others or cannot be done without other agents. There are different application fields, such as traffic, airports or logistics. Here, patterns for reuse in the system are presented, which can be seen more generally and used in different domains for similar problems. Due to different designs and problem fields, grouping patterns which can solve problems are useful. Furthermore, the patterns help the developer by providing questions and answers. The most suitable pattern regarding the answers is investigated.

Mechanisms of self-organization like cooperative behavior are valuable because agents can be organized into configurations for useful applications without imposing external centralized controls.

5.4.1 Goals

An individual or group has a clear goal that is consistent with the mission and vision. It allows all members to select the means by which they achieve their goals autonomously. Clear goals enable a group to measure its progress, making decisions and avoiding conflicts. Without clear goals, a group has problems in solving its tasks.

In order to accomplish its goal, the work the group performs is defined as a group task. Group members must be autonomous in accomplishing the task and share a collective responsibility for the group's performance and result.

Platooning is discussed with various suggestions like truck platooning, a lead truck and vehicles following, or vehicles only, and for different behaviors such as, for instance, following or dedicated lanes. Whereas the term 'platoon' results from traffic control aspects, this thesis uses the term 'groups', indicating that they evolve from a purely decentralized perspective - the individual autonomous vehicles. The goal of aggregating vehicles and minimizing gaps as done in AHS is the same in platoons or groups.

Can possible grouping strategies answer the question For what groups are needed? A group or team comprises people or agents linked by a common purpose. For conducting tasks that are high in complexity and have many interdependent subtasks teams are especially appropriate.

1. Possible grouping for functional requirements are:
    - to gather input information
    - to classify functional areas of the system
    - to cluster outputs
2. Incomplete list of grouping information:
    - By same kind i.e., personal car, public bus, transporter, truck
    - By purpose i.e., light signals, message signals, moving objects/vehicles
    - By common goals i.e., same route, green phase passing, same next steps/decisions
    - By organizations/companies i.e., taxi company, public transport
3. Agent coordination on different levels:
    - goals
    - plans
    - actions
4. Therefore, group formation on different levels:
    - for goals based on destination node and/or area, destination coordinates need to be compared.
    - for plans or partial plans, matrices are compared for similarities.
    - for actions, the agent's most recent actions on the environment ? i.e., driving straight, turning left or right, accelerating or braking.

5.4.2 Motivating Task

According to `Hackman` [150], a motivating group task meets certain conditions:

- Members have significant autonomy over how they accomplish the work, so that they feel the ownership of their work.
- Members are enabled to use their variety of skills.
- Working on the task generates feedback to members about their performance.
- The whole and meaningful work has a visible outcome.
- The outcome has significant consequences for others, like traffic management and executive bodies.

For group effectiveness, the members need to meet several criteria:

- For completing the task successfully, the members should have the required knowledge and skills.
- The group size should adapt to the task, because more members means more time is needed for coordination.
- The group composition needs to be stable to maintain the continuity of effort, but still be flexible for new or differing ideas.

Group behavior has been studied by many disciplines like Artificial Intelligence (AI), robotics, biology and sociology. This work attempts to contribute to group behavior by the synthesis of the different disciplines AI, traffic and ethology.

5.4.3 Group Mission

A group's **mission** answers the question of the reason for its existence, and a *vision* is a mental picture that a group seeks to reach in the future. The vision includes what the group should look like and how it should act to accomplish its mission. Both mission and vision are used by the members to guide their work, and they should be able to express them.

Cooperation is seen in this context as an engine for self-organization. It is assumed that individual vehicles will reach their goals faster in a group. The cooperative attitude of an agent is threefold (cf. [272]): local, independent of the global function of the system, and heuristic to move through state space in the right direction.

An agent is cooperative if:

- c_{per} perceived signals are comprehensive
- c_{dec} the agent interprets the received information without ambiguity
- c_{act} competent actions are useful for other agents

Thus, the proscriptive approach is that agents must avoid or resolve non-cooperative situations (NCS): $\neg c_{per}$ or $\neg c_{dec}$ or $\neg c_{act}$.

Definition 5.4 *Let G be a group 5.5. A system $\mathcal{X}$ of elements of G is called a generating system of G or a system of generators of G if the smallest subgroup of G containing $\mathcal{X}$ is equal to G, i.e., every element of G is expressible as a product of the elements of $\mathcal{X}$ and their inverses. (...) If $\mathcal{X}$ is a generating system of the group G and $\mathcal{R}$ a corresponding system of defining relators, then $\langle \mathcal{X} \mid \mathcal{R} \rangle$ is called a presentation of G and indicated by this writing $G = \langle \mathcal{X} \mid \mathcal{R} \rangle$. (...) A group G is called finitely generated if it has a finite system of generators and finitely presentable (or presented) if it has a presentation with a finite number of generators and defining relators*[2].

Initializing

- give each vehicle its own coordinate system (initializing digital map)

[2] [69] p. 6f.

- each vehicles senses its relative position and orientation to others (requirement for group formation)
- in purely decentralized systems, no superior exists
- each vehicle has a relative position feedback - therefore vehicles are stabilized [384] robots symmetrically and proven mathematically
- a two-dimensional "formation vector" makes the formation controllable by vectors - refer to computer simulations

5.4.4 Group Membership

A member is an agent of a group, sharing technical competence, professional knowledge and allocation of resources in order to maximize its utility. Members usually have no conflict of interest in fulfilling a goal together at the time of group work. Members are characterized by their group work and are flexible in following a goal, trust each other and interact with other agents.

According to `Hackman` ([150] p. 324), group members work in a task-oriented fashion if it is motivationally engaging (for instance, their utility improves), the organizational reward system provides challenging performance objectives and reinforces their achievement, and there is positive interaction among members for shared commitment to the group and its work.

Membership and the task for group work are tightly interlinked. The design of the group should meet the following conditions:

- Members should be able to use their skills.
- Tasks should be complete and meaningful and have a visible outcome.
- The outcome of the group's work on the task has significant consequences.
- Substantial autonomy is required for the way group work is done.
- The group's performance feedback is given back to the members.

If a group meets these criteria, then members engage and experience their work as meaningful. Members feel collectively responsible and work on the performance outcome. This creates motivated group members and group synergy with minimized process losses for coordination and motivation, as well as shared commitment to the group and its work.

Effective group members give sufficient effort to the group task, integrate their knowledge and skills and develop good performance strategies.

Members as well as groups can be non-cooperative or cooperative. Noncooperation is out of scope for this thesis and cooperation is explicitly demanded.

Cooperation is a main concept in multi agent system specifications. In cooperation with each other, the agents can manage to work effectively and to achieve collective goals in complex and dynamic environments. Therefore this concept is essential for improving efficiency and effectiveness and is very important for the design of distributed solution systems.

Models can be distinguished by their form of time progression, which are the known model paradigms of continuous and event-based simulation. Continuous or time-discrete models are calculated in each simulation cycle. Most macroscopic models are represented by differential equations which are iteratively calculated. For this purpose, the time progression is adjustable at will, but must be discrete.

Event-based simulation is oriented on the actual changes with time progression. An event causes a change, which is subsequently propagated by the model. Newly arising events are sorted into an event chain which sorts every event by its appearance time. After that, the simulation tick is switched to the next time step, when the next event is scheduled. Those models can be represented by Petri nets.

This thesis assumes the continuous time-discrete model, because traffic simulations are generally time-discrete.

The duration of a simulation step is influenced on three levels: by the model, by the interpreter and by the hardware. The model is the base described here, the interpreter is presented in Chapter 6, and the hardware in Chapter 7.

Finally, two kinds of time are mandatory for a group to complete its tasks and achieve its goals: performance and capacity-building time. During performance time, the group works towards achieving its goal and, during capacity-building time, the group tries to optimize its performance. Examples are redesigning a work flow to increase efficiency or avoid conflicts and improving the group's abilities with more information.

5.5 Group Formation Process

To improve traffic throughput, vehicles can collaborate by driving in groups. A general vehicle group is a subset of the existing vehicles all driving on the same segment at the same speed, but not necessarily in the same lane of the segment.

Definition 5.5 (Group) *Let $(N, E, V, C, contr, cap)$ be a traffic system. Then $g \subseteq \{(v_1, sz_1, rp_1, (s_{1,0}, s_{1,1}, l_1), sp_1^{act}), (v_2, sz_2, rp_2, (s_{2,0}, s_{2,1}, l_2), sp_2^{act}), ..., (v_n, sz_n, rp_n, (s_{n,0}, s_{n,1}, l_n), sp_n^{act}) | \forall_{1 \leq i \leq n-1} s_{i,0} = s_{i+1,0}, s_{i,1} = s_{i+1,1}, sp_i^{act} = sp_{i+1}^{act}\}$ is a group of vehicles and G is the set of all such groups.*

For simplification, each vehicle is driving in exactly one group at any point in time, i.e., the set of vehicles V can be partitioned by a set of groups

$$\overline{g} \subseteq G$$

.

Definition 5.6 (Set of groups) *Let $((N, E), V, C, contr, cap)$ be a traffic system and G be the set of groups of TS. Then $\overline{g} \subseteq G$ is a partitioning of V iff $\forall_{v \in \biguplus \overline{g}} Z(v) = 1$ and $\biguplus \overline{g} = V$. $\overline{G}$ is the set of all such partitions of V.*

Vehicle Groups

For this thesis, the group of autonomous vehicles G^X is defined as follows:

Definition 5.7 (Vehicle Group (G^X)) *Function $f : X \times Y \to R$, described in the following 5.5, calculates the dissimilarity between two autonomous vehicles X and Y. G^X is a vehicle group and X is the group leader only if:*

$$\forall Y \in G \ f(X, Y) \leq \alpha$$

The definition 5.7 specifies a vehicle X as the group leader of the vehicle group G^X. Any arbitrary vehicle will be allocated to the group G^X, if the dissimilarity between its characteristics and the characteristics of the group leader X are smaller or equal to the α value. That implies that vehicles with the same characteristics form a group, so that vehicles of two groups have different characteristics. A member of a group is an autonomous vehicle which has the characteristics of the mean properties of all members in the group, so that the properties of the group are also the characteristics of the member of the group. It is important that the properties of two groups can be compared.

For example, the desired speed of group A and group B can be compared to determine which group is faster. In theory, the comparison of the characteristics of two groups can also be done by the members of those corresponding groups. There are two methods to define the characteristics of a group member:

- The first method defines a virtual member. The member has corresponding characteristics to the mean characteristics of all members of its group.
- The second method defines the group leader as the group representative.

The amount of members of a group changes permanently in a dynamic traffic environment. In case the first method is implemented, then the characteristics of the representative or all group members need to be updated incessantly. This might cause a calculation complexity with overhead for a dynamic environment like urban traffic in which the autonomous vehicles need to decide quickly, although for decentralized agents it would be the preferred method. However, the first method does not seem very applicable to the urban environment. Thus, the second method is used throughout this work, while being aware of creating a small hierarchy. The dissimilarity value of the second method is in the range of $[0, \alpha)$ for the group leader and a virtual representative of the group (member).

Weighting Function

The weighting function 5.22 of the dissimilarity derived from the 'Manhattan distance' with the impact of the desired speed ds, acceleration acc and deceleration dcc characteristics, denoted with α values, is defined as follows:

$$f(X, Y) = \alpha_1 \frac{|ds_x - ds_y|}{s_{ds}} + \alpha_2 \frac{|acc_x - acc_y|}{s_{acc}} + \alpha_3 \frac{|dcc_x - dcc_y|}{s_{dcc}} \tag{5.22}$$

The following condition 5.23 for $\alpha_1, \alpha_2, \alpha_3$ needs to hold, so that the characteristics do not alter the meaning of the α value:

$$\alpha_1 + \alpha_2 + \alpha_3 = \alpha \tag{5.23}$$

The weighting function 5.22 has two important properties: robustness and flexibility, which means in detail:

- Robustness: the weighting function needs to be robust with regard to changes in the environment. That means if a vehicle Y, at the time step t, is a member of group G of vehicle X, then, also at time step $t+1$,vehicle Y is a member of the same group G of vehicle X. Thus, the decision of vehicle X whether the vehicle Y can take part in its group is only dependent on the static characteristics of vehicle Y and the definition of acceptable deviation, not on the changes of the traffic environment.
- Flexibility: the weighting function needs to be flexible with regard to the needs of the autonomous vehicle (or the driver). This implies that vehicle X needs to be able to influence the weighting function $f(X,Y)$ in order to find the desired members. By inserting the new $\alpha_1, \alpha_2, \alpha_3$ values into the function $f(X,Y)$, it becomes more flexible. The vehicle X can adjust the fitting $\alpha_1, \alpha_2, \alpha_3$ values for its group itself and therefore change the amount of potential members dynamically.

Manhattan Distance Function [3]

$$f(x,y) = \alpha_1 \frac{|desiredspeed_x - desiredspeed_y|}{s_{desiredspeed}} \tag{5.24}$$

$$+ \alpha_2 \frac{|acceleration_x - acceleration_y|}{s_{acceleration}} \tag{5.25}$$

$$+ \alpha_3 \frac{|decceleration_x - decceleration_y|}{s_{decceleration}} \tag{5.26}$$

$$+ \alpha_4 \frac{|routecost_x - routecost_y|}{s_{routecost}} \tag{5.27}$$

$$+ \alpha_5 \frac{|sumroute_x - sumroute_y|}{s_{sumroute}} \tag{5.28}$$

$$\tag{5.29}$$

$$\sum_{i=1}^{n} \alpha_i \cdot w_i$$

[3]The following three paragraphs were collaboratively devised by Jana Görmer, Thomas Hornoff, Philipp Kraus, Christoph Kuper and Thomas Plathe in 2014.

$$\alpha_i \cdot w_i := \text{edge id times costs of weights i.e., time, fuel consumption,} CO_2$$

$$routecost := \sum_{i=1}^{5} \alpha_i = 1$$

$$prim_i = f(i)\text{implement as function to be able to exchange prime numbers}$$

$$sumroute := \sum_{i=1}^{n} prim_i^{id}$$

$$id := edge$$

The Manhattan norm generates the Manhattan distance. Between corresponding coordinates the distance is the sum of the differences between any two points (or vectors).

Adapted Euclidean Distance Function

$$\begin{aligned}
f(x,y) &= \alpha_1 \cdot ||desiredspeed_x - desiredspeed_y||_2 &(5.30)\\
&+ \alpha_2 \cdot ||acceleration_x - acceleration_y||_2 &(5.31)\\
&+ \alpha_3 \cdot ||decceleration_x - decceleration_y)||_2 &(5.32)\\
&+ \alpha_4 \cdot ||route_x - route_y||_F &(5.33)\\
&+ \alpha_5 \cdot ||routecost_x - routecost_y||_F &(5.34)\\
&+ \alpha_6 \cdot ||position_x - position_y||_2 &(5.35)\\
&+ \alpha_7 \cdot ||destination_x - destination_y||_2 &(5.36)\\
& &(5.37)
\end{aligned}$$

Frobenius norm: $|| \cdot ||_F := \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (X_{i,j})^2}$
with $\sum_i \alpha_i = 1$

Similarity Algorithm Based on the AEDF distance function, the following steps need to be executed.

1. calculate distances between pairs of vehicles
2. make the distance matrix symmetrical with $\frac{1}{2} \cdot (A + A^t)$
3. sort the values of the upper and lower triangle matrix
4. determine the median of those values
5. all vehicles which have a distance value smaller than or equal to the median form a group
6. maybe optimize α_i

The distances are defined by the sum of the weighted difference of the characteristics:

$$\alpha_i \cdot ||\text{characteristic of A} - \text{characteristic of B}||_2 \tag{5.38}$$

Every characteristic is an element of the metric vector space such as the space to represent speed, a space for routes, etc. For that reason, a norm must be defined which determines the “distance“ between two elements.

On that base, similarities (which are usually specified as dissimilarities) between two vehicles (or more general elements) are calculated. For that, another distance function is necessary, which exists for the pseudo metric space. The definition of a metric space has the following criteria:

Non-negative: *if the distance between two elements is 0, it results in the same element* $d(x, y) = 0 \Rightarrow x = y$ additionally true for $d(x, y) \geq 0$

Symmetry: *the distance of x to y is identical to the distance f y to x* $d(x, y) = d(y, x)$

Triangle inequality: *when having three elements x,y,z, then the direct path between x and y has at least the length of x to z and of z to y* $d(x, y) \leq d(x, z)d(z, y)$

When calculating similarities, the criteria of non-negative drops out, because the distance of a similarity can be 0, even though it is between two different non-identical elements: $d(x, y) = 0 \not\Rightarrow x = y$

The weights of α_i designate how much the characteristic influences the similarity values. It would be best if the α_i were optimized during run time, but this is not in the scope of this thesis, although definitely desirable for future research. Due to the definition of $\sum_i alpha_i = 1$, it is a convex optimization problem. It requires a definition of which criteria the alpha values can be changed for. The alphas can be interpreted in such a way that group formation is influenced by single factors. The alpha values determine on which factor groups form, have group stability and decompose. Combinatoric and numeric methods can be used as optimizing procedures.

The similarity algorithm presented here is used for decentralized dynamic vehicle groups in MATI, the final version of this thesis after investigating different algorithms.

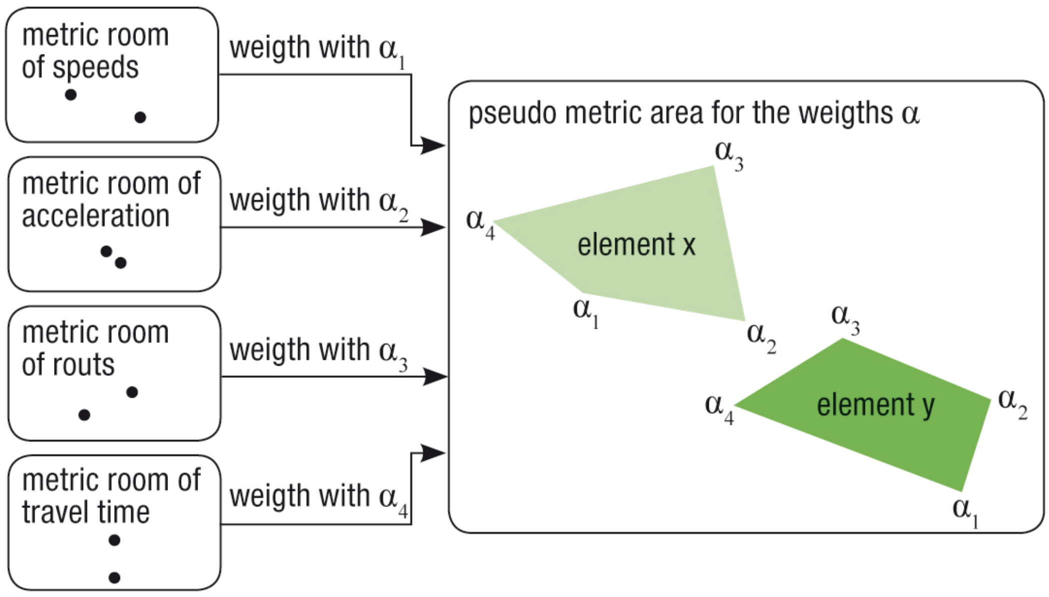


Figure 5.10: Vectors.

Group Stages

In group theory according to `Tuckman and Jensen` [350], five phases are distinguished: forming, storming, norming, performing, and adjourning; these are illustrated in the Background 2.1 and addressed in the following paragraphs. For this thesis, three phases for groupings are defined, which are then explained.

Forming During the Forming stage of group development, members are usually excited to be part of the group and optimistic about their future work, sometimes combined with anxiety about fitting in and their performance within the group. Asking a lot of questions is a common behavior by group members in the Forming stage, reflecting both their excitement about the new group and the uncertainty they might be feeling about their place in the group. The principal task for the group during the Forming stage is to create a group with a clear structure, goals, direction and roles. Based on that, members begin to build trust. A good orientation process defines the group's mission and goals. Then, the group's expectations about both their work and its result and, more importantly, the group's process, need to be communicated. However, during the Forming stage, much of the group's energy is focused on defining the group, so task accomplishment may be relatively low.

One of the combinatorial algorithms is an algorithm of searching - finding a particular type of equivalent classes - to find the group. It uses 'Search by scanning': all edges are scanned, and those joining vertices's with the strength of the bond exceeding a definite threshold are combined. During an exhaustive search with recursive enhancement of all group input, all possible unions, intersections, and complementation of the sets are obtained. The final group input forms a field (F). Using the focus of attention we receive a special category of limited fields which do not contain all the combinations of components required by a formal definition of the field.

One option of group formation is, if each vehicle is per definition always part of a group, group forming means that vehicles change groups, i.e., the partition of V changes to another partition of V:

Definition 5.8 (Change Groups) *Let* $((N, E), V, C, contr, cap)$ *be a traffic system and* $\overline{G}$ *be the set of all partitions of* V *into vehicle groups. Then* $change \in \overline{G} \to \overline{G}$ *is a function that changes the group assignment of individual vehicles.*

Note that a vehicle can only change to another group when it is driving on the same segment of the traffic system as the group it is joining. Thus, the vehicle has to adjust its position and speed before it is able to join the group.

A vehicle decides to change groups whenever joining another group is beneficial for the vehicle. Hence, the vehicle requests information from vehicle groups close to its current location (but not necessarily in exactly the same location) and evaluates them.

Definition 5.9 (Information about Groups) *Let* $TS = ((N, E), V, C, contr, cap)$ *be a traffic system with* RP *as the set of route preferences and* G *as the set of groups of* TS*. Then*

- $getLocation \in G \to E$ *provides the actual position of a group,*
- $getActSpeed \in G \to \mathbb{N}$ *provides the actual speed of a group,*
- $getRP \in G \to RP$ *provides the route preference of a group,*
- *...*

Storming During the Storming stage, members are trying to see how the group will respond to differences and how it will handle conflict. As the group begins to move towards its goals, members discover that their early excitement and expectations cannot all be met. Their focus may shift from the tasks to anger with the group's progress or process if they feel unable to meet the group's goals. Frustration or disagreements about goals, expectations, roles and responsibilities are openly expressed during the storming stage. Group members may argue or become critical of the group's original mission or goals. Group tasks refocus the group on its goals, which can be portioned into achievable steps. A redefinition of the group's goals, roles and tasks might be helpful. The group needs to develop both the overall group process including conflict management skills and individual task-related skills.

From the agent perspective, the Storming stage decides roles and properties of the agents, which is done by the design and modeling of agents. This phase is regarded as minorly important and included in the preparation phase, which combines the first three phases of `Tuckman` [349] [350]. The only decision agents have is whether to join the group, depending on their utility function and preferences, or to decline the group goal and mission.

Norming During the Norming stage of group development, members begin to adapt to conflicting expectations and create a group reality for proceeding with the group process. If the group is successful in setting more flexible and inclusive norms, then accepting others in the group leads to increased group cohesion where + expectations of members are met. Exchanging information and recognizing the variety of opinions and experiences makes the group stronger and its result richer. During the Norming stage members make a conscious effort to resolve problems and achieve group harmony. There might be more frequent and more meaningful communication among group members, and an increased willingness to share ideas or ask group members for help. Members refocus on established group ground-rules and practices and focus on the group's tasks. Thus, they shift their energy to the group's goals and for individual and collective work the productivity increases. The group may find that this is an appropriate time for an evaluation of group processes and productivity for effective joint work. The agent design needs to set rules and the criteria for groups, as well as group benefits.

Performing In the Performing stage of group development, members feel attached to the group as something 'greater than the sum of its parts', and the group's effectiveness gives satisfaction. They share insights into personal and group processes and are aware of their own (and each other's) strengths and weaknesses. Members feel confident in their individual and group abilities. Group members are able to prevent or solve problems with the group's process or progress. A 'can do' attitude is visible in assisting one another. Members take on various roles and responsibilities as needed whereas differences among members are appreciated and used to enhance the group's performance. In the Performing stage, the group makes significant progress towards its goals. Commitment to the group's mission is high and the competence of group members is also high. Members should improve constantly their knowledge and skills, and continue their group development. Accomplishments in the group's process or progress are measured and celebrated. Any changes can cause a group to cycle back to an earlier stage like members joining or leaving as well as large-scale changes in the external environment. If these changes - and their resulting behaviors - are recognized and addressed directly, groups may successfully remain in the Performing stage - even indefinitely.

In the context of this thesis, this is the main phase where agents work together toward their goal. This can be done with swarm algorithms for static and predefined systems and the Manhattan distance for dynamic platooning. After sorting the vehicles into groups and dedicated lanes, they perform their behavior of driving in the group lane.

Adjourning Groups dissolve when their work is completed or when the organization changes. `Tuckman's` original model [349] did not include the Adjourning stage, but it is important for any group to pay attention to the end or termination process.

Group members can achieve contentedness from the accomplishments of the group. Individual members might feel sadness or a sense of loss due to the changes to their group relationships. Given these conflicting feelings, individual and group morale may rise or fall throughout the ending stage. During the Ending Stage, some group members may become less focused on the group's tasks, but return back to their individual priorities. Transitions and the different individual group feelings about the group's impending dissolution should be acknowledged. During this stage, the group should focus on three tasks:

1. Any deliverables should be completed and the remaining group work finished
2. Identifying 'lessons learned' with an assessment of the group's process and result, respecting and using them future groups
3. Acknowledgment of the individual contributions and the achievements of the group with a closing celebration that formally ends the group's existence for this purpose.

Adjourning is the opposite of the Forming stage; the changing of groups includes adjourning. After group behavior and rules, the individual priorities and utilities become more important again. Vehicle groups need to end their group phase of traveling together in the post phase because they will park for shopping, living or working.

Predefined Phases The three predefined phases are oriented to and inspired by `Tuckman` [350], but simplified into the three following phases:

1. Preparation phase for group formation: forming, storming and norming are combined. Forming, in which the groups are established and the whole process is quite insecure, storming, in which roles are defined and properties are negotiated, and norming, which in static groupings is the most important preparation phase in which group rules, advantages and group criteria are set.
2. Main phase for group performing: working toward the group plan.
3. Post phase for adjourning: the group dissolves either into subgroups or to individual behavior.

It is important to predefine how long the phases need to be and when each phase ends.

The **preliminary phase** is dedicated to group formation. At the beginning of a simulation, all vehicles are in the preliminary phase, meaning that they are performing independently and are trying to build a group with other vehicles. Once they are in a group (or have decided not to be in a group)the main phase of the simulation starts.

The preliminary phase uses centralized color sort algorithms and, decentralized, the adapted Manhattan Distance Function and/or the Euclidean Distance Function (explained before in Subsection 5.5). Two triggers for the end of the phase can be identified: either there is a predefined fixed end phase or it depends on when the autonomous vehicles end the phase. The decentralized approach is more challenging and can include communication and coordination overhead, which needs to be respected. It is assumed that, when automatizing especially decentralized group formation, in most cases the individual reaction time, the vehicle properties, the communication and the coordination determine the preliminary phase. A measure known for each vehicle is the time span of braking distance, which is set equivalent in order to identify similarities to other neighboring vehicles.

Reaction time is defined as the product of the speed and the perception-reaction time of the driver. The vehicle properties original speed of the vehicle and the friction coefficient are respected when braking and therefore also when identifying similarities to neighbors.

The total stopping distance is the sum of the perception-reaction distance and the braking distance, and, for the determination of the prephase, the braking time is multiplied by the number of neighbors.

$$\begin{aligned} Distance_{total} &= Distance_{perception-reaction} + Distance_{braking} \\ Distance_{prephase} &= (v \cdot t_{p-r} + \frac{v^2}{2\mu g}) \cdot number_{neighbors} \end{aligned} \tag{5.39}$$

with the values incorporating the ability of the vast majority of drivers under normal road conditions.

$t_{p-r} = 1.5s$	s seconds	v initial driving speed
$\mu = 0.7$	μ coefficient of friction	g gravity of Earth

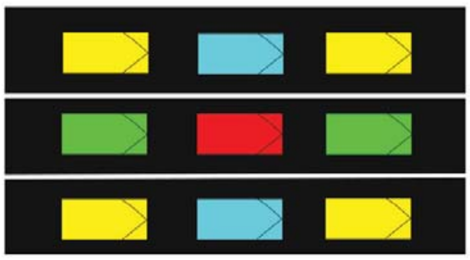


Figure 5.11: Neighboring vehicles.

The number of neighboring vehicles depends also on the amount of lanes, but the three lanes illustrated in Figure 5.11 show *eight* direct neighboring possibilities from the perspective of the red vehicle in the center position.

- In the same middle lane, the red center vehicle has *two* neighbors, green in color: one in front and one following.
- The red center vehicle has to its sides *two* blue neighbors in neighboring lanes.

- The red center vehicle has **four** diagonal neighbors in yellow in neighboring lanes.

Other variations of neighbors are possible:

- one: direct front (or rear) neighbor.
- two: either in other direct lanes (vertically) the right or left neighbor or, in the same lane, (horizontally) predecessor or follower.
- three: the three front (or rear) vehicles
- four: the combination of horizontal and vertical neighbors.
- five: the surrounding vehicles aiming in one direction, front and vertical vehicles.
- six: the neighbors effected by car following and lane changing, such as five vehicles plus the one following in the same lane.
- seven: the direct neighbors except the one in front (or rear)
- eight: all direct neighbors diagonally, vertically and horizontally.

Considering vehicle groups in the preliminary phase, similarities should be checked in a certain time period, starting from when the vehicles enter the simulation and finishing when group formation is finished or at a finishing line. The end to the process of group forming can be given by environmental contexts like an intersection or another predefined line (i.e., by dividing the scenario into three equivalent sections). For both triggers for the end of the preliminary phase, it is effective to check similarities with only the front three neighbors or, additionally, including the side neighbors, which makes five neighbors in total. This way, not all eight neighbors need to be checked, but just three (or five). If the vehicle is not in the center, but on a peripheral lane, just two (or three) neighbors need to be compared for similarities. Thus, it is assumed that at least three neighboring vehicles should be compared in order to identify similarities and therefore accept or decline being part of the group. At the most it will take the braking time combined with the number of neighbors which need to be compared.

The **main phase** is dedicated to the group performing the simulated group behavior. As soon as all vehicles are either in a group or have decided to drive independently, the main phase of the simulation starts. Group behavior by the vehicles is desired in order to maximize the individual performance and the overall flow of the traffic simulation. In the main phase, the groups drive with a group speed and can use less safety distance than when driving individually, which should enhance the group's driving performance in contrast to individual vehicles. Vehicle group coordination can be performed on different levels, as presented in Subsection 5.1.1:

- *goals* are based on destination line, node and/or area, therefore i.e., destination Global Positioning System (GPS) coordinates need to be compared. This happens in centralized vehicle groups in which a destination line is given until which the vehicles drive jointly.

- *plan* or partial plan matrices are compared for similarities. This is what happens in the decentralized main phase.
- *actions*: the agent acts on the most recent environment ? i.e., driving straight, turning left or right, accelerating or braking. This happens for each individual vehicle, but also for group actions.

The centralized group is predefined by the sorting before and occupies one group lane at a steady speed. It is estimated that this is the most effective group driving behavior. The decentralized groups are more dynamic because, also with the MDF/EDF, group members can act like a swarm of fish using the full capacity of the lanes but can also regroup in the main phase.

The **post-phase** is dedicated to the adjourning of the group and reestablishing the individual vehicle behavior. The moment vehicles leave a group and do not join another (i.e., because they are going to park at their homes or offices), the vehicle enters the post phase. In that phase, the vehicles are approaching their final destinations. In simulations, the vehicles generally leave into a centroid.

5.5.1 Conflict Management

Conflicts are a natural part of the group life cycle and usually improve the member's ability to work together and accomplish their task as well as contributing to individual growth. A conflict provokes an information exchange and therefore provides the ability to learn more about other viewpoints or approaches to the task. Reflecting (and learning) about the conflict and resolving it in a way that it stays resolved, and thus learning to act differently to prevent unnecessary conflict, are influencing factors for effective groups.

Group conflict can be separated into two sub-categories: *inter-group* conflict, in which distinct groups of individuals are at odds with one another i.e., resource conflicts, and *intra-group* conflict, in which select individuals that are part of the same group collide with one another i.e., role conflicts.

Group Conflicts

A group conflict is a situation between vehicle groups with limited resources. In the following, a resource conflict regarding limited lanes is depicted in which one group is driving fast and the other slow, It is defined in the following definition 5.10.

Definition 5.10 (Conflict between Groups (G^{V_i} and G^{V_j})) *$X_V(t)$ is the position of a vehicle X at the time step t. The vehicles V_i and V_j are the group leaders of the groups $G_i^{V_i}$ and $G_j^{V_j}$ respectively. Different conflict situations and the relative positions of the groups are defined as follows:*

- *$G_i^{V_i}$ is behind $G_j^{V_j}$ $(h(G_i, G_j))$ if:*
  for all $V_e \in G_i$ and for all $V_f \in G_j$ holds $X_{V_e}(t) < X_{V_f}(t)$

- *$G_i^{V_i}$ it in front of $G_j^{V_j}$ $(v(G_i, G_j))$ if:* *for all $V_e \in G_i$ and for all $\in G_f$ holds $X_{V_e}(t) > X_{V_f}(t)$*
- *$G_i^{V_i}$ and $G_j^{V_j}$ overlap $(u(G_i, G_j))$ if:* *$\exists V_e \in G_i, \exists V_f \in G_j$ such that $X_{V_e}(t) \geq X_{V_f}(t)$ and $\exists V_n \in G_i, \exists V_m \in G_j$ so that $X_{V_n}(t) \leq X_{V_m}(t)$*
- *$G_i^{V_i}$ and $G_j^{V_j}$ are in a conflict $Conf(G_i, G_j)$ if:* *$h(G_i, G_j) \vee u(G_i, G_j)$ and $\exists V_i \in G_i, \exists V_j \in G_j$ so that $min(f(V_i, V_j), f(V_j, V_i)) > \alpha$, whereas $f(V_i, V_j)$ and $f(V_j, V_i)$ are calculated with the help of function 5.22. The value of α is 3. $ds_i - ds_j$ is bigger than s_{ds} of V_i. ds_i and ds_j are the desired speeds of vehicles V_i and V_j.*

A group conflict can be detected by the leader or by members. This means, all members x_a of a vehicle group G_x maintain the information of their group. Every member of a group is a conflict detector and receives the characteristics of the group leader. As soon as a member detects a member of another group in range of perception, it asks the other member for its group leader characteristics. At each simulation step, a member x_a sends a message $msg(id, id_y, id_x, reqGinfor)$ to its neighbors z to find other groups. Receiving the neighbors response, the member x_a uses the following binary decision function to determine whether there is conflict between its own group G_x and the neighboring group G_z.

$$conf(G_x, G_z) = \begin{cases} yes & \text{if } ds_x - ds_z > w_{ds,x} \wedge \\ & \exists z_n \in G_z, p_{z_n} < p_{x_a} \\ no & \text{otherwise} \end{cases} \tag{5.40}$$

where p_{x_a} and p_{z_n} are positions of x_a and z_n on the street. Once a conflict is detected $(conf(G_x, G_z))$, the member forwards the information about it x_a and sends the information of the conflict group G_z to its own group leader. Based on the answer, the group leader knows whether it is in conflict with the other group or not. With the coordination of *group lanes* group leader manages its members. Group lanes are reserved for members of a group. The choice of group lanes bases on the following two criteria:

1. Lane changing of members should be minimized by group lanes.
2. The group lanes should assure that fast groups are not blocked by slow groups.

Global Coordination

Global coordination is a form of strategic planning and aims to solve conflicts between vehicle groups. The global coordination consists of agreements between group representatives (the leaders) of conflicting groups about the use of different lanes. Those agreements are reached by negotiation. Negotiation [113, 253] is a mechanism which allows autonomous agents to reach agreements on a topic of interest.

This mechanism is used to resolve conflicts or to allocate tasks among multiple agents. Detailed reading is presented in `Müller` ([253] p. 97f). For the purpose of this thesis of evaluating autonomous vehicle groups in urban traffic, protocols for task allocation like the classical contract net protocol were used, and others are discussed in Section 3.5.4.

The group leader communicates the group lanes to its group members. Driving in the group lanes can prevent being blocked by other members of a group. Every group leader receives a priority, which is connected to its desired speed: the bigger the desired speed, the higher its priority. The group leader with the highest priority can choose its group lane first. Since there are no preferences given by the group leader, there are three possible methods [319], dominance, maximin (pessimist) or maximax (optimist), for technical weight assignment from the field of game theory. There are multiple criteria for the decision support, but a simple method, in which the solution is reached individually in the strategic decision, is the dominance method. The principle of dominance states that if the strategy of a vehicle leader dominates over another leader in all characteristics (here just the desired speed is taken into account), then the latter strategy is ignored because it will not effect the solution in any way. Thus, the lane choice is made using the dominance method. The group leader calculates the dominance value for every lane x of his group G by using this function:

$$Do_x(G,S) = Num_x(G) - \sum_{G_a \in S} Num_x(G_a) \tag{5.41}$$

where S are the conflicting groups of the group G and the group leader of S is slower than the group leader of G. If a lane is not chosen by any group leader, it will be marked as 'free'. The group leader X_i of G chooses all free lanes with the dominance values $Do_x(G,S) \geq 0$ as his group lanes and marks them as 'occupied'. In case no lane is free or there is no lane with the dominance value $Do_x(G,S) \geq 0$ then the group leader X_i chooses the lane with the biggest dominance value.

This process could be done by negotiation [113, 253] or market-based approaches like auctions [362], but this is out of scope for this thesis.

5.5.2 Communication

Communication is embedded in all group processes of the group life cycle. Therefore, the sender and receiver need to understand the meaning of the message in the same way. Group interaction is a broad descriptor which also incorporates respect, fairness, and support, which need to be defined and discussed on the level of individual and collective behavior (represented in Figure 5.3 after the group context) so that they are consistent with the desired (group) values, assumptions, and norms.

Specific of message protocols are required for communication between vehicles. The Car2Car [277] communication consortium intends to find a European standard for vehicular and infrastructure communication in traffic. Cooperative

Awareness Messages were used in the PLANETS project [110, 137]. The European standardization [180] uses two message types: event-based messages and status information messages. Each participating vehicle informs its receivers frequently about their environment and various applications are supported. Those messages are called Cooperative Awareness Messages (CAMs). A CAM is sent only when a set of criteria is met so that the resulting CAM sending frequency is in the range of 1 to 10 Hz. Local dynamic map (LDM) aggregate the CAMs in a database. Information about the traffic situation can be extracted and used for vehicular and infrastructure communication in urban traffic.

For agent-based communication, a simple message format in the form of

$$\texttt{msg(id,}id_s\texttt{,}id_r\texttt{,content)}$$

was defined for all messages exchanged between vehicles, here `id` is the identifier number of the message. id_s and id_r are the identifier numbers of sender and receiver respectively. 'Content of message' contains all exchanged information between vehicles.

According to a BDI architecture (cf. [285]), the ICL and SCL are each divided into three segments. *Belief* the world state as it is perceived. In the SCL, this state is restricted by the organization into functional, structural and deontic specifications. In the ICL, it is regulated by the behavior of other agents and the environment conditions. *Desires* are identified as goals of single agents (individual context) and of agent groups (social context). *Intention* comprises executable plans for fulfilling individual and/or joint goals.

5.5.3 Boundary Management

For this thesis, the main boundary is traffic management, which provides the information and the tasks, whereas tasks and decisions are also made with the help of multi-agent systems. Groups ensure that traffic control provides the materials (the street network with traffic lights and signs), technologies (Road Side Units, Signal Plans, Variable Message Signs etc.) and information (traffic information of road works etc.) which is needed for accomplishing its tasks. Traffic management has a static influence due to its own complexity. For future work this could be more dynamic.

Depending on the task options created by the environment infrastructure, for example, turning left or right at an intersection or traffic light, the agent decides which task is the best for reaching the goal point in the traffic network. Hence, multi-agent simulation helps autonomous vehicles in their decision making process and performing tasks in the environment. The biggest advantage of multi-agent simulation ([206] p. 81) is the underlying concept, which is very close to the modeled original system. That means the individual, which represents reality, can also be represented in the model. Especially when replicating societies like urban traffic, which are composed of autonomous entities, the modeling process is more direct than with an equivalent macro model. The emphasis of multi-agent systems is modeling individual behavior, which is formed by natural observation for data acquisition and then validated. For example, an individual vehicle is marked and its actions followed in its urban traffic environment or its reactions observed towards changing situations like a road blocked by construction or a traffic jam. In general, the behavioral model of an individual agent is not limited in its formulation, just by performance issues for the simulation.

In addition to the direct mapping of individuals, inhomogeneities in the environment as well as heterogeneous societies can be represented. Thus, emergent behavior can be investigated and feedback mechanisms illustrated in different system levels. Inherent modularity is another advantage of multi-agent simulation.

The simulation of a multi-agent model within traffic simulations poses some problems ([206] p. 84). The biggest drawback of every individual-based simulation is the amount of resources needed compared to the abstract model of the system on the macro level. There is a high demand on the calculation and storage capacities of the computer used: every individual needs an equivalent in the computer and needs to be actualized separately. Therefore, the amount of simulated individuals is crucial and the minimum number needs to be found. A possible solution is to simplify the agent behavior and use other abstractions like probability distributions. The question is, which amount of agents is mandatory and how many are sufficient?

Often individual systems simulate a few agents whereas big populations are simulated with macro models with many agents, where the individual characteristics statistically are no longer relevant. Here a mesoscopic model could solve the problem.

Interactions between agents are very complex to express in a formula without ignoring important effects. Simple techniques like game theory models can be used for rational agents for representing and analyzing. This is in n contrast to the combination of entities, which need to react in local situations and adapt to the dynamics of a changing environment, and are best modeled with interacting agents.

Differentiation of Groups

All participants of traffic could be divided into groups for the technical and environmental side, consisting of signals, traffic lights, VMS, etc., and mobile participants, such as vehicles by attributes (fast, slow, big, small), by according categories (sport car, truck, bus, regular) or other properties. The group identity is important for the coherence and separation of groups. If groups use the same resources, for instance the same street segment, then a group conflict of how it is used by whom could occur. Solutions are provided in the conflict management 5.5.1 section, and inter-group behavior in which slow groups block fast groups, is investigated mainly in the ATSim approach 4.3.4.

It is important for the designer to differentiate groups using the right criteria, find a mode of conflict or prevention of conflict, as well as to declare conditions of failure of conflict resolution. Equal roles which are similar, qualitatively distinct, or interdependent should be respected and rewarded accordingly. Inter-group concerns are fostered through goal structure, division of labor, or mutual role indispensability. Groups need to be equally valued for their intergroup roles.

Criteria on which to differentiate groups could be their behavior with cooperative or competitive strategies. This thesis assumes cooperative urban traffic, which means the behavior is benevolent towards other groups, a global strategy and individuals. Competitive group behavior could be, for example, blocking other vehicles for group benefit or sacrificing a vehicle to block others from driving in the same direction like in the game of Lemmings. Other strategies could be using communication to achieve advantages which could play with the trust of others. Since it is a very broad and complex field, competitive groups are not in the scope of this thesis.

The characteristics of groups can be homogeneous or heterogeneous, defined by the group member properties. Both are investigated in this thesis, with groups of the same vehicle type, which is possible in simulation, and different vehicle characteristics in form of length and speed behavior, so that the results are visible in simulation. Since traffic is mostly individual nowadays, this is also considered for research, where the simulation starts with individual vehicle behavior, then static or dynamic groups are formed and ending with individual behavior. This best reflects reality, where vehicles are used for a purpose which could be improved collectively like saving time and, at the end, the vehicle parks at its destination. In dynamic groups, group merging depending on similarities is possible and is done automatically.

External Integration

For development and experimentation, four tools can be distinguished. The modeling interface constructs agent specifications and then the environment with graphs and programming, which is often combined with the simulator. The simulator interprets the model in a step-based manner and illustrates changes for the actual model configuration where the data can be logged for protocol data. For this thesis it is an agent-oriented traffic simulator. Often it includes the animation component with different representations of the data i.e., logged or visualized data. The animation component uses the data from the simulator to display the model configuration. Experiments can be viewed in different configurable views. Finally, the evaluation component is used for the interpretation of the experiments. The data can be analyzed on the base of the simulator protocols, the processes of values of the state variables, the amount of agents etc.

The agent behavior needs to be integrated into the external traffic system, where interfaces need to be used and, often, the systems run on different programming languages. Often traffic systems are programmed in C++, whereas the agent software is often JAVA-based. Therefore, the application programming interface (API) needs to be used. This could have drawbacks because not all functions are interlinked, like in SUMO with the TraCI4J interface, where functions for JAVA which are available in C++ need to be added. If the simulation software is commercial, like AIMSUN, then the company needs to be asked to provide add-ons to support agent behavior for vehicles. This could save valuable implementation time, but usually takes longer and software communication problems could occur. When two simulations like AIMSUN and JADE are combined, then a communication bottleneck with two simulations waiting for each other could interfere with the performance. Parallel agent execution could probably alleviate the problem, but currently the agent languages are mostly object-oriented and not parallel.

5.6 Implementation: Grouping Algorithm

The implementation (cf. [206] p. 47) is an executable model for simulation experiments. The definition of a certain modeling paradigm or a concrete formalism is necessary. The preferred procedure consists of, firstly, defining the modeling structure sufficiently exactly and, subsequently, depending on the test simulation runs, calibrating the model until the desired behavior is reproduced sufficiently. With multi-agent models, the adjustment refers to several considerations: the behavior of the modeled individual as well as the global behavior should correlate with the point of reference.

Discussion of Group Formation Algorithms

The group formation algorithms need to take into consideration group goals, autonomous vehicles, and existing infrastructure as described in the State of the Art, with the assumptions outlined in Section 3.6.

A cooperative urban traffic network is the underlying field where autonomous vehicles make their rational choices (for more information see Game Theory), but are only partially benevolent (for more information see Cooperative Distributed Problem Solving), since the agents are self-interested in that they maximize their utility function. Joint and individual utility functions are described in Definition 2.1.

The criteria for the groups which are formed are stability, duration, and the following informal parameters:

- *heuristic* versus exact
- how good are the algorithms regarding performance, complexity and fulfilling their purpose?
- what is the goal function for the group solution?
- which parameters are changeable?

Centralized Static (CS+CNP+DS) The idea behind the centralized approach is to use something simple and straightforward. First, the vehicles are sorted with a color sort algorithm, then grouped into leader and members with the role-based Contract Net Algorithm, and, finally, group behavior is aborted and the group dissolves back to individual behavior.

The sorting of the vehicles does not need to be unique, but depending on the vehicle classes, which are represented by a color, this sorting is continuous, which is good for the traffic context. Color sorting can lead to very different results. Thus, two color sorting approaches were pursued: from the start line forward and from the finish line backward. In a way, the same segments of the prephase are used for vehicle sorting.

The sorting was done with three colors for three lanes, but was found to be very static, because the occurring events are hard-coded. This did not seem feasible for the dynamic and changing environment, where three lanes are not always present. Therefore, random vehicles are generated into the net. The vehicles located near by compare each other and sort themselves by their characteristics. The first vehicles in line determine which group lanes are chosen. The vehicles indicate when they plan to change their group lane and wait for the gap to be big enough to change lanes assuming that the traffic flows as normal. Done the backward way, the sorting process will always finish before the next phase. This is recommended for the static process, because new incoming vehicles are sorted once they enter the net and the next phase, the group driving, can start immediately without waiting for the sorting to be finished.

The contract net protocol is used for the group driving. Through the sorting, the first car in each lane at the start line of the grouping phase will be the group leader and all the other cars will take on the preferences and behavior of the leading car. This way, they drive in a very coordinated fashion, very likely in one lane without changes. This leads to a near optimal speed for the group leader. It is also possible to adjust the gaps between the members so they are smaller, which is one benefit of vehicle groups.

In the last phase, after the starting line for the post-phase, the vehicles leave the group and resume their original driving preferences. At this point the difference between coordinated and normal driving can be demonstrated.

The results of the static group sorting are very predictable, but therefore not very realistic.

Decentralized Dynamic (AEDF) Groups The defining attribute of multi-agent systems is the interaction between agents, which encompasses planning problems in a decentralized setting, learning other agent models, composing groups with high task performance, and selecting resource-bound communication and coordination. The significant variety of used methodologies for such problems includes symbolic reasoning about negotiation and argumentation, distributed optimization methods, machine learning methods such as multi-agent reinforcement learning, etc. Some form of prior coordination is needed for the majority of these well-studied methods. Often, the coordination is at the level of problem definition. For example, learning algorithms usually assume a common learning method or prior beliefs shared by all agents. Or distributed optimization methods may assume specific structural constraints regarding the partition of state space or cost/rewards, and symbolic methods often make strong assumptions regarding norms and protocols. However, in realistic problems, these assumptions are easily violated, calling for new models and algorithms that specifically address the case of multi-agent interaction without prior coordination. Similar issues are also becoming increasingly more pertinent in human-machine interactions, where there is a need for intelligent adaptive behavior, and assumptions regarding prior knowledge and communication are problematic.

This dynamic group formation algorithm aims for a mature adaptation of the Euclidean Distance Function (AEDF) related to multi-agent interaction without prior coordination. This includes theoretical (described here in the following) and empirical (described later in Subsection **??**) investigations of issues arising from assumptions regarding prior coordination in interactive settings, as well as solutions in the form of novel models and algorithms for effective multi-agent interaction without prior coordination.

This algorithm includes:

- Agent coordination and cooperation without prior coordination
- Event-based interpreter data including beliefs, goals, planning and coalitions according to the Beliefs Desires and Intentions (Model) (BDI) paradigm
- Group formation (and information sharing) in ad hoc settings
- Vehicle agent interaction without prior coordination

Autonomous Vehicle Groups

In creating autonomous vehicle groups, the behavior of the system as well as the behavior of the vehicle agent when interacting with the urban traffic system are defined. All vehicles are autonomous, therefore no human drivers are modeled or included in the simulation.

System Behavior For autonomous vehicle groups, which are introduced here, the system allows a vehicle agent to retrieve information which is relevant for its next actions. This includes signal plans and the rest time values of the signal phases. Based on this information, the vehicle can calculate its behavior and make its decisions.

In the example of `Dresner` [89], individual vehicles can reserve the spaces where they plan to go, so that the intersection can be divided into an

$$n \times n$$

grid of reservation tiles, where n is the granularity of the reservation system. Thus, each tile can be reserved by one vehicle per time step. The vehicle sends a message containing several parameters for reservations.

1. The time the vehicle will arrive.
2. The velocity at the vehicles arrival.
3. The direction at the vehicles arrival.
4. The vehicle's maximal velocity.

5. The vehicle's maximal and minimal acceleration.
6. The vehicle's length and width.

The intersection simulates the journey of the vehicle including these parameters in order to forecast and recording cells occupied by the vehicle at each time step (as well as a few time steps before and after, for safety). If any of these cells are already reserved, the intersection rejects the driver's request. Otherwise, it accepts the driver's request. This grid-based approach does not fit entirely with this thesis' continuous space approach and needs to be adapted for future use.

Another future add-on feature is that traffic lights can guarantee the green phases to vehicle groups. Also, communication is essential and, similar to the parameter messages for reservation, the traffic light needs to receive all relevant vehicle group information.

Vehicle Group Behavior At the beginning of every cycle, the vehicle agent determines its own route depending on its position, including the actual global weights for the street segments received via infrastructure communication from the global control center for strategic decisions. Then it makes its own tactical decisions as to which lanes it will use and how fast it will be driving.

If the agent has not yet been included in a group, it sends a message with its individual data of actual position, route and desired speed. If a group accepts the request due to a similarity check, the driver agent notes that joining a group is possible along with the parameters and can accept the group plan. If the driver has joined, the group parameters are reevaluated. If it determines that it cannot meet the group parameters, it leaves the group and adopts its individual parameters. If the group rejects the request, the driver initiates its own group at the next time step and other vehicles can join. After three time steps without another member in the group, the group is canceled, so that no vehicle is alone in a group. The group formation process begins again.

Every agent compares itself with other agents including itself, because individual agent parameters are used for the group. That way, the agent can leave the group and set its own parameters. Thus, the 'HashBasedTable' is filled with similar values which are also logged in the distables.

Group Formation Policies

Using the MATI simulator, the performance of two different group formation policies, static and dynamic vehicle groups, have been evaluated with four experimental scenarios. Additional investigation was done on the influence of homogeneous and heterogeneous characteristics of vehicles as described in the agent design of Section 6.3.

The general simulation execution cycle is described in the following listing 5.6, indicating that it can start one of the four given scenarios with a maximum of 5275 vehicles, which is the average number of vehicles of the Hanover Südstadt with a traffic density LOS D. Each vehicle is initialized with an agent in the environment network. For each simulation step, the information of network and interpreter is updated wherever vehicles enter, leave or perform an action in the network.

```
At each simulation cycle:

Start

    scenarios = [3_lane, hildesheimer, grid, h_sued]

    for i=0 to 5275
        for each scenario in scenarios do
            initialize the vehicles in a given network
            initialize each vehicle with an agent
        end for
    end for

    for simulation step=0 to the end of the simulation step
        update_network_information(step);
    end for

End

Func update_network_information(simulation_step)
Start

    // add vehicle
    if (simulation_step == the time step a vehicle should be added)
            add vehicle to the network
    end if

    // calc next action
    if (vehicle in the network)
        calculate its next action and update it
    end if

    // remove vehicle
    if (vehicle arrived at its destination)
        remove this vehicle
    end if

End
```

All this data is logged in the 'tripinfo.xml' file by SUMO.

Static The static is by far the simplest of the four policies in terms of group formation: vehicles strictly follow the algorithms of the three phases: the pre-phase is a sorting algorithm, then the main phase is maintain speed and lane in the group, and the post-phase adopts individual driving as the priority action.

More explicit modeling and design needed to be done for the static group formation in order to indicate where the phases start and end, so traffic lights were integrated into the environment. Therefore, the given architecture supports the static group formation process, so that results can be achieved. The length of the phases is a point of interest and could be modified, but is out of scope for this thesis. The optimization ability is investigated in many approaches as stated in the State of the Art Chapter 3. The only delays are caused by vehicles traveling in the same direction.

Dynamic Dynamic group formation feels natural, because vehicles join groups in their vicinity if they have equal or very similar characteristics, i.e., maximal or desired speed, acceleration, deceleration, actual position, destination and route. The individual vehicle compares its parameters with the group and sends a request to join if they are equal or similar. Depending on the factor of dissimilarity and the amount of group members, the group can accept or decline the potential member. If the member joins the group, it adopts the group behavior as a priority action base, but still stores its individual beliefs. Thus, it is possible for the member to leave the group at any time in case it determines that the parameters have changed significantly when performing internal checks.

The environment can also have static and dynamic elements which influence the grouping process. The dynamic group formation model is a close approximation of vehicle behavior on real streets. Some extra information for the vehicle agents is helpful for them to decide on their actions based on traffic lights, which are static, but changing objects are often seen as agents (but not within scope of this thesis).

The model has two pieces of information: the traffic light phases and its duration for all four directions. Note that opposing lights are always identical. Yellow lights are not necessary for agents, since they react instantly. In real-world traffic signals, yellow lights alert the drivers that the light is due to change to red. However, in the dynamic group model, the traffic light phases can actually be queried by the vehicles to determine at any time in which state the light will be. This helps the vehicles conducting through the intersection and could give guarantees of green phases for vehicle groups in future research.

The interaction between the vehicle agent and the traffic light is straightforward. First, depending on its current velocity the vehicle calculates the arrival to the next intersection with its corresponding traffic light. The vehicle then sends a message to the intersection informing it of the time at which the autonomous vehicle expects to arrive (which refers to Infrastructure to vehicle communication, like C2I or V2I (I2V) communication). The intersection then responds with the range of times during or after the time specified by the vehicle at which the lights will be green (which refers to Vehicle-to-Infrastructure Communication, same as C2I (V2I) communication). The vehicle agent can then accelerate or decelerate as necessary to ensure that the vehicle enters the intersection when the light is green.

It is assumed that vehicles can notice changes in their environment, such as position changes of neighboring vehicles, only in a certain range of perception. This assumption derives from the technologies used, i.e. wireless LAN like Wi-Fi, Bluetooth, Wimax or radio waves, which can be used by autonomous vehicles to perceive the environment. Group forming needs to be an instinctive behavior of every autonomous vehicle to enforce vehicle groups. Thus, at every time, vehicles are motivated to find a suitable neighbor for group formation. The following Algorithm 2 (compare [64] p. 56) uses the weighting function 5.22 to form an autonomous vehicle group around a vehicle Y. Input: individual autonomous vehicle driving alone. Output: autonomous vehicles driving in a group with a leader.

Note, that 'direct neighbors' in row 2 stands for all vehicles which are in the vicinity of vehicle Y. The algorithm 2 contains two phases.

In the first phase, the vehicle Y wants to join a group and requests neighboring vehicles for information of their group leaders in the following message format $msg(id, id_y, id_x, reqGinform)$, where id is the identifier number of the

Algorithm 2 Algorithm for group formation: participating in an existing group and creating a new group

```
1: $Y$ is in no group (free vehicle).
2: $Y$ sends requests to its *direct* neighbors for group leader information
3: For every group leader $X$ found, $Y$ calculates the value of function $f(Y,X)$
   and adds $f(Y,X) \leq \alpha_Y$ to a list $L_1$ and the corresponding leader $X$ to
   another list $L_2$.
4: while $L_1$ is not empty do
5:     vehicle $Y$ requests the group leader $X \in L2$ for participation, which
   dissimilarity value $f(X,Y) \in L_1$ is smallest.
6:     The group leader $X$ confirms 'yes', if the function fulfills $f(X,Y) < \alpha_X$,
   otherwise 'no'.
7:     if the answer of the group leader $X$ is 'yes' then
8:         vehicle $Y$ commits to group $G^X$ and stops the group formation process.
9:     else
10:        vehicle $Y$ erases group leader $X$ from $L_2$ and the corresponding value
   $f(Y,X)$ from $L_1$
11:    end if
12: end while
13: if vehicle $Y$ is in no group and a direct neighbor $Z_i$ exists (or more direct
   neighbors), such that $f(Y,Z_i) \leq \alpha_Y$ and vehicle $Y$ has the smallest ID
   number and $Z_i$ belongs to no group then
14:    vehicle $Y$ creates a group $G^Y$. $Y$ is in group $G^Y$
15: end if
```

message, and id_y and id_x are sender and receiver. The content of the message contains information exchanged by the vehicles. A member of a group only has the static characteristics of the group leader, so a grouped neighboring vehicle of the requesting vehicle Y can always send its group leader information. The vehicle Y uses the dissimilarity function 5.22 to filter the group leaders with whom the vehicle Y can connect. The vehicle Y chooses the most similar group leader and sends a request of participation $msg(id, id_y, id_x, reqPar)$ to the group leader X . This group leader X uses the weighting function with the variable alphas 5.22 in order to decide whether the requesting vehicle Y can participate in its group. In case of a positive answer $msg(id, id_y, id_x, yes)$ by the group leader X, the group formation process terminates successfully.

An additional second phase is started if the vehicle Y gets a negative response. In this case the vehicle Y checks for possibilities of creating its own group. In doing so, the vehicle Y evaluates its surrounding vehicles. In case the vehicle Y finds a neighboring vehicle Z_i with similar attributes to form a group, then the vehicle Y creates its group G^Y and waits for a participation request from the neighboring vehicle Z_i. The group creator takes the role of group leader. However, the following conditions apply when a new group needs to be created:

1. the vehicle Y is not a member of any group,

2. the vehicle Y knows of at least one candidate vehicle Z_i with $f(Y, Z_i) < \alpha_y$,

3. the vehicle Y has the greatest id in comparison with the id of other candidate vehicles.

The vehicle Y cannot guarantee that the neighboring vehicle Z_i wants to take part in its group G^Y. The group leader monitors the number of members and decides when it wants to remove its group. After a defined time (three simulation steps in the case study), the group will be deleted by the vehicle Y if the group G^Y is still without other members, because maintaining an empty group does not allow the group leader to participate to other groups.

Due to the limited range of perception, sometimes the vehicle Y cannot connect directly to the group leader. In row 5, an indirect request of vehicle Y to the desired group leader X is described. This results from the assumption that every member of a group can take the function of a router (multi-hop communication) and every member can communicate with all the other members of its group including the group leader. Because the vehicle Y is a member of the group G^X and connected to it, therefore, the vehicle Y can also communicate indirectly with the leader X. For the later implementation, this is simplified to direct communication, assuming that Y can send a direct request to the group leader X.

5.7 Summary

This thesis aims to change the focus of software assistance from 'classic centralized' to 'autonomous individual' with the help of mesoscopic vehicle groups for decentralized coordination in cooperative urban environments. The focus of this chapter was a model for vehicle group formation, enabling the systematic use of models as primary engineering artifacts throughout the group life cycle.

This chapter modeled decentralized dynamic vehicle grouping algorithms, conflict detection and global coordination methods. The model describes two autonomic features: a centralized as well as a decentralized system. First, a predesigned *centralized* model using three phases was introduced: sorting as a pre-phase, grouping as the main phase and dissolving as the post-phase. Second, a *decentralized* approach to autonomic grouping based on a dynamic multi-agent coordination algorithm in urban traffic was designed, where the agents represent the vehicles and form groups, taking vehicle properties information (maximal and desired speed, acceleration and deceleration as well as the current location and destination as well as their route) into account.

Future traffic management only makes sense using the synergy of traffic control, communication and automated vehicles as a connected package. Dynamic and decentralized traffic management should receive a higher priority, with the goals of standardization and an information gathering market [270].

Intelligence is the ability to adapt to change.
Stephen Hawking (1942)

Chapter 6

Urban Traffic Groups: A Case Study

Autonomous vehicles dynamically form ad-hoc groups; driving in such a group enables them to drive with a common speed and a smaller gap to the predecessor, thus increasing traffic flow which benefits the individual as well as the whole traffic system. The main research question is whether models and methods of autonomous group formation and coordination can be applied in the context of cooperative decentralized traffic management.

In this chapter, the approach is to design the decentralized and centralized organization and cooperation of traffic participants, the autonomic vehicles, in order to form groups. Introduced in chapter 5, describing the model, and shown in Figure 5.1, this influences five areas:the design of (1) the simulation including (2) the environment with (3) the agents and (4) their interaction, and last, but most important for this thesis, (5) the organization. The simulation component, acting as the testbed for real life, has the biggest influencing factor, which is why all the other components (2-5) are described on the base of how the simulation is done. This is described in the following Section 6.1.

The structure of this chapter is based on the simulation components. Therefore, creating the environment - the base for the simulation - is described first. Then the agents, how they act in the environment and their mobile properties are introduced. The interaction features are sometimes included in the agent paradigm, but due to their importance also for grouping autonomous vehicles a whole section is dedicated to them. The last topic, although most important for this thesis, is the organization of urban traffic groups, its algorithms and structure, which can also be included in the environment. Thus, the simulation can also be broken down to its environment with organizational features, and agents facilitating interactions, which is done in MATI. This is the simulation base for this thesis, described in the following Section 6.1. In Chapter 7, the constructed and described use cases, in the form of two realistic scenarios, are compared and evaluated with methods of decentralized dynamic versus centralized static models of group formation.

6.1 Simulation Scenarios

In this thesis, cooperative decentralized traffic management is defined as the process of observation and optimizing the traffic flow in the net(-work). This especially pertains to the autonomy of traffic participants as agents and the communication between them with Vehicle-to-X technologies. The use of multi-agent methods and models for group formation and coordination is particularly investigated regarding the requirements and limitations of the decentralized traffic management. The scenarios represent different environmental contexts for vehicle group formation in cooperative traffic and are equivalent to the case studies.

Possible use cases are specified in the following based on the agent traffic requirements and constraints and the model in chapter 5. The group life cycle for different use cases is instantiated. The similarities of the local and global goal functions of traffic management plays an important role. Aspects of quality from the local view and for the entire system are to be respected in form of travel and stop times. Thus, it is possible to manage such a decentralized system with goals of traffic management, that is energy minimization including reducing emissions. At present there is no existing group formation which routes targeted and selective traffic flows individually or in a convoy of vehicles and, therefore, it is an extension of the traffic management employed nowadays. First, the settings for all scenario case studies are presented in the following Section 6.1.1, and then the scenarios are itemized from simple to complex networks.

6.1.1 Settings

For this thesis traffic is apportioned into mobile autonomous vehicles and static infrastructure elements like traffic lights. All vehicles are represented as agents and equipped with an intelligent on-board unit including autonomous driving kit, communication and navigation. In a first approach, vehicles are multiple 'cars' of the same kind and infrastructure is 'traffic lights', and in future including 'detectors' and 'variable message signs'. Infrastructure elements are managed by the microscopic traffic simulators like AIMSUN and SUMO with enhanced expert knowledge of Dynamic Traffic Management[1].

Basic assumptions for the case studies are:

1. all scenarios are to be tested with randomized initial distributions of at least 100 simulated vehicles in three different simulations *without groups*, *with centralized groups* and *with decentralized groups*.

[1]Notable help was provided by Daniel Schmidt `https://www.tu-braunschweig.de/ivs/institut/mitarbeiter/ehemalige/schmidt` of the Institute of Transportation and Urban Engineering, Technische Universität Braunschweig, Brunswick, Germany.

2. all scenarios are to be divided into *three predefined phases*: the pre-phase, the main phase and the post-phase for centralized static group formation.
3. the goal is to *minimize the weights on the edges* (here for each lane), i.e., travel time, shortest/fastest route, CO_2 emissions, fuel consumption.
4. the scenarios are to be tested with *homogeneous and heterogeneous autonomous vehicle types* including the drivers' behavior.

The assumptions are described in detail in the following.

Randomization

For the first assumption, the simulation runs need to be randomized in order to obtain statistically significant statements. The Appendix C.2.1 describes an example of a randomized implementation.

Phases

For this thesis, three phases for groupings were defined in the model chapter 5.5: first, the pre-phase for group formation, second, the main phase for group performing and, third, the post-phase for adjourning. It is important to predefine how long the phases need to be and when each phase ends. This was done using natural borders in a local scenario as follows: before the traffic light is considered the pre-phase, within the traffic light the main phase, and, after the last traffic light, before leaving the network, is considered the post phase. But phases could also include several intersections and need to be defined with the scenario settings.

Different Simulation Runs

Each scenario runs in three different simulations *without groups* as default, *with centralized groups* and *with decentralized groups*. First, the scenario is run without groups to see the basic setting of the simulation. In order to evaluate the benefit of vehicle groups, two simulation settings with groups are tested: the global perspective with centralized static groups through an observer/controller model and the individual view with decentralized groups where the vehicle agents can be seen as peers.

Homogeneous and Heterogeneous Vehicle Types

The fourth assumption regarding homogeneous and heterogeneous autonomous vehicle types is described in detail in the following Subsection 6.3. The drivers' behavior is included in the simulation environment using driver models of reaction times and variances.

6.1.2 Artificial Three Lane Scenario

The three lane scenario is the most basic for traffic scenarios. This can be seen as a highway connecting two cities or a long road in an urban environment (the latter is the focus of this thesis). The road is 3 km long and each phase is therefore 1 km long, which is suitable for urban traffic.

Possible actions are: acceleration, braking (seen in Figure 6.1 in the first yellow vehicle) and reaching the desired speed, involving car following and lane changes in order to drive right, left or in the middle. Also, simple group formation for centralized static and decentralized dynamic algorithms can be tested and simple experiments conducted. Thus, the microscopic behavior of vehicle groups is simulated including how long the three phases take; this can then be used in the other scenarios.

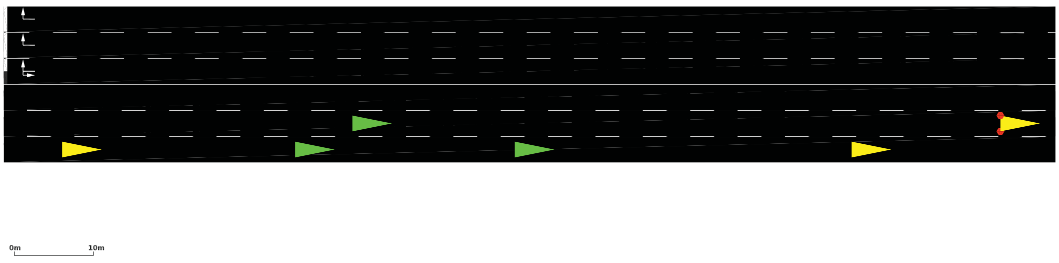


Figure 6.1: 3-Lanes created with SUMO Traffic Simulation.

In the following Section 6.2, the three lane scenario is described in detail as created with SUMO. This scenario was also used in Streetworld 4.3.1 and, with more detail, in ATSim 4.3.4.

6.1.3 The Intersection Scenario

The intersection scenario is used for investigating vehicle groups with three specifications which are without, with centralized compulsory and with decentralized groups.

An intersection is a street junction where two or more streets either meet or cross at the same level. There are 3-way intersection, also known as T junction or fork, the most common 4-way intersections like crossroads, and rarely 5-way or more. It may often be controlled by traffic lights, or it can be a roundabout.

Most intersections are 4-way intersections with a crossing over two streets (or roads), which are perpendicular to each other, but may cross at different angles. Often those intersections are regulated by traffic lights considering main traffic flows. For example, during the morning rush hour, incoming traffic streams from South to North whereas the West and East streets have lower priority. Turns are usually allowed, but in order to avoid interference or collision with other traffic they are regulated by regulatory signs or signals depending on flow limitations or restricted streets such as one-way streets. Considering right-side driving countries like Germany, left turns should be made from the leftmost lane and right turns from the rightmost lane to avoid collisions or blocking of traffic going straight. Generally, U-turns are not allowed. Turning lanes increase the capacity of an intersection and improve safety according to `Harwood et al.` [155].

The intersection scenario modeled by SUMO as shown in Figure 6.2 is a 4-way intersection with two perpendicular crossing streets, each having three lanes for easy regulation of turns. It is an intersection of rather large proportions of turning traffic, which is why turn lanes are provided where the street approaches. There are left and right turn lanes for traffic approaching the intersection in westbound and eastbound directions. Right turns can be performed without crossing traffic whereas left turns and U-turns are often allowed and regulated by the traffic light. Usually, traffic control signs are used to prohibit turns or when there is a lack of a signal phase, i.e., for left turns.

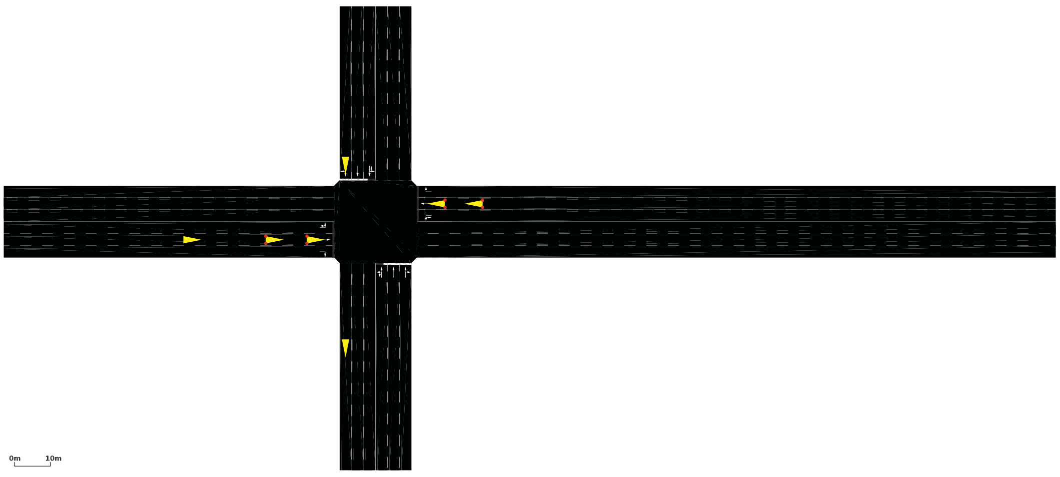


Figure 6.2: 3-Lane Intersection.

From the perspective of this thesis, the research question of this intersection scenario is: how can the management be more autonomous and more decentralized? What are the possible influencing mechanisms? What is the maximum amount of decentralization?

For simulating realistic behavior, this intersection scenario is partly managed by a central institution for infrastructure elements like traffic lights, detectors and variable message signs, and decentralized mechanisms are provided by the behavior of vehicles in the form of the agent paradigm illustrated in Figure 6.3.

The scenario has an intersection with a 'traffic light' (in Urban Engineering it is called a 'signal group') on each side. In total there are four visual traffic lights with two signal groups: North-South and East-West. Traffic lights are managed with actions of green, red and yellow phases, and times are scheduled for normal traffic behavior with opposite lights with the same signals at a central place, the traffic management.

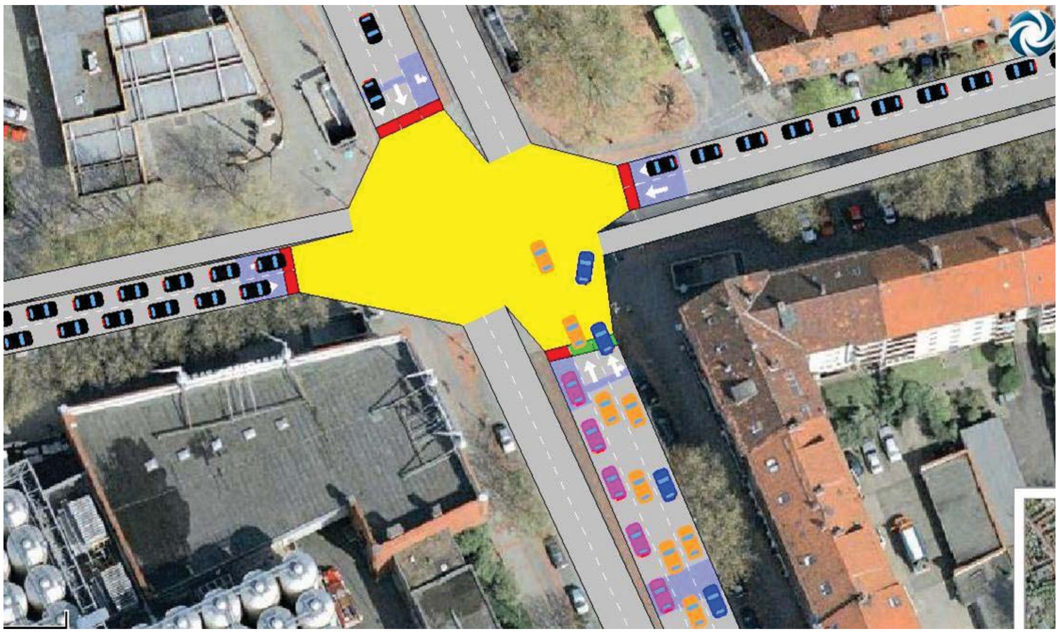


Figure 6.3: Intersection with Vehicle Groups for each direction modeled with AIMSUN.

For realistic behavior of vehicles, they are managed by the traffic simulator with car following models and pre-implemented behavior by AIMSUN or SUMO as default. Additional behavior of learning and grouping derives from agent modeling as an add-on to the traffic simulation. This thesis attempts to adopt (new) agent behaviors: Grouping individual vehicles and dynamic vehicle groups.

The focus of the intersection scenario is grouping for common goals like the origin and destination of the individual route plan. Vehicle groups are established in a compulsory fashion at a red traffic light in order to have coordinated group behavior in the next green phase. Thus, the following parameters are needed for the autonomous vehicle agents:

- Traffic management: the infrastructure recommendation of preferred routes resulting from the overall network (using an efficient algorithm like Dijkstra for routing of platoons), where and when traffic density (Level of Service C or D) could trigger vehicle groups for a certain spatial area, and the local perception of the traffic light color with rest time value which includes the agent actions: go, stop, slow down, speed up.
- agent behavior: default, individual agent decisions, group decisions, agent sensors for perceiving the environment, effectors for actions in the environment, and internal what to store?
- communication of infrastructure to vehicles in general for recommendations, group communication between members and group to infrastructure communication.

- algorithms for network routing and traffic density to trigger grouping and efficient group coordination algorithms.
- what are the minimum and the optimal distance between vehicles? This results in a recommendation i.e., direction choice: next action go left, go straight, go right and decide optimal and possible lanes (later routing) to fulfill requirements, be aware of side effects (other vehicles, infrastructure elements, groups?). Decisions can be made for direction and possible lanes for the chosen direction based on data of total number of vehicles, number of vehicles in same direction, estimated travel time from this lane into this direction from previous numbers or Floating Car Data (FCD). Future possibilities are to take more complex Peer-to-Peer data and grouping decisions into account.
- an optional time-layer with the visual/mathematical zones green, red, critical, can support an agent to make a decision. For example,to violate some soft rules in order to group like 'speed up more than given speed limit', 'go when it is yellow/red?', 'slow down to wait for next phase'. Also the maximum and optimal numbers of vehicles for one group including all side parameters given by the infrastructure can be respected. Detectors can support to answer the questions 'how many vehicles are in the Section before the traffic light at present?', 'How many can form a green-phase group','Do other groups from other directions exist?' and finally 'What are priorities for grouping vehicles?'

Agent actions are for the sending and receiving of communication, and can also react to the environment with the movements *go or accelerate*, *stop or decelerate* or *turn*. Actions regarding grouping involve joining a group, accepting, rejecting, no answer, confirmation and acting in a group (driving in a chunk with minimum distances if possible) as well as dissolving into individual vehicle behavior.

The intersection scenario is also included in the following green wave and grid scenarios. In order to see all group phases, more than just one intersection is needed. Thus, this intersection scenario is not used for the evaluation, but was needed as a miniature scenario to test whether all the technical functionalities of the different research groups of PLANETS can be implemented.

6.1.4 Realistic Hildesheimer Street Data as a Green Wave Scenario

For the green wave scenario, realistic data from March 2009 of Hildesheimer street, starting at Altenbeker Damm and ending at Aegidientorplatz, Hanover, Germany, is used. This green wave scenario coordinates several intersections in one main South to North route with mostly two lanes in each direction, illustrated in Figure 6.4. This scenario allows a continuous traffic flow over the green wave which is calculated by traffic engineers who decide on an approximate speed where 50 km per hour is allowed, but, in general, the Level of Service is between C and E, indicating moderate to heavy traffic involving slowing. Due to a progressive cascade of green lights which are timed statically, vehicles traveling along with the green wave do not have to stop at intersections. This is beneficial for the individual vehicle and the overall network. Less acceleration and braking is needed and, therefore, the green wave allows higher traffic loads, and reduces noise and energy use. Thus, the quality of life in urban areas is improved.

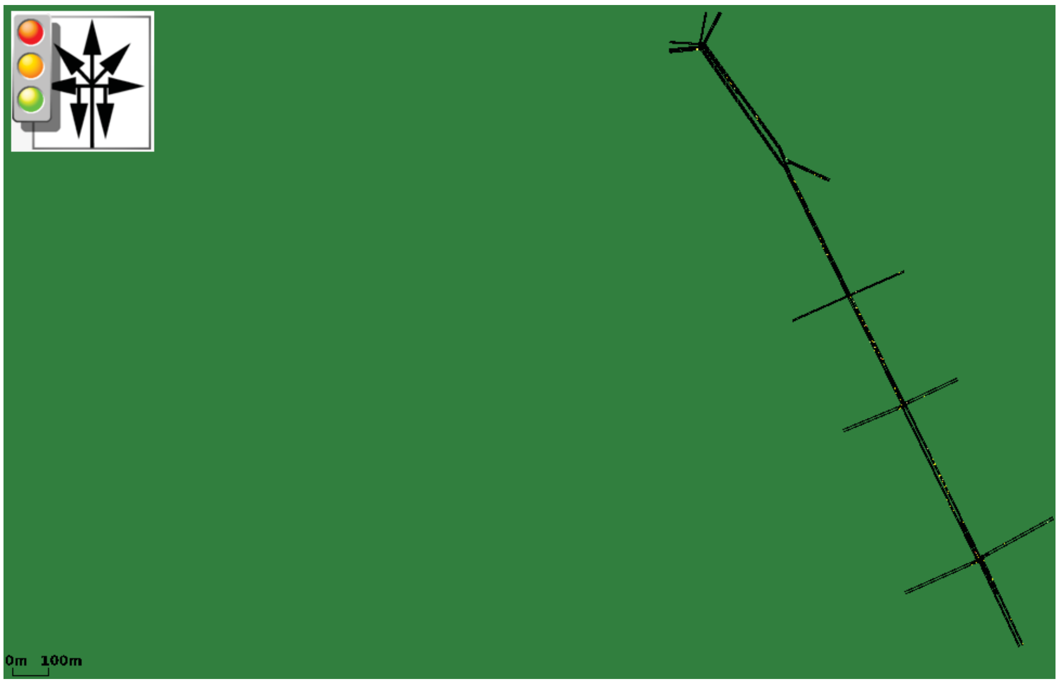


Figure 6.4: Hildesheimer Scenario as a Green Wave modeled with SUMO.

In practical use (compare `Fiosins et al.` [109]), a group of vehicles, called a 'platoon' by traffic engineers, can use the green wave. The platoon is limited in size by the signal times where different time phases give way to other traffic flows. The phases are defined such that the first intersection ends the segment before it as the preliminary phase, the main phase starts with the vehicle crossing the intersection at the green light and ends two intersections later, and the post phase starts before the last intersection so that the individual behavior can be seen.

In this scenario, the group effect is taken further and it is investigated whether centralized or decentralized groups can improve the traffic load from the global and individual perspective. At the first intersection, vehicles with similarities are grouped compulsorily for dynamically for decentralized traffic groups and then the joint vehicles drive together with using all lanes possible (versus a group lane). In order to see all phases, more than just one intersection is needed.

The objective of the scenario focuses on grouping throughout a sequence of regulated intersections. The vehicles intend to pass several successive intersections without stops for minimizing their individual travel times. In the network optimal throughput can be regulated through traffic lights accordingly and is favored by the centralized traffic management. The idea is to create a motivation for vehicles to join groups and act in a coordinated fashion by informing them of reduced travel time i.e., 'green wave' priority, which is present in public city transport. Thus, beneficial interaction between the centralized and the decentralized approach is created. Further research can include guarantees for agreements i.e., of green phases, by the traffic lights for vehicle groups or a certain amount of individual vehicles. This would include a two way communication from bottom-up V2I and top-down like most practiced I2V nowadays.

For coordinated actions and their goals, groups in this thesis' simulation are (re-)formed at red traffic lights for centralized groups; they then cross the following intersections as a platoon. In the 'green wave scenario', the first vehicle arriving at the intersection acts as a group leader. Then the group process is triggered and a prediction of the traffic state and of the signal plans of subsequent intersections is made by the leader. Depending on the outcome, the leader creates and broadcasts the group plan consisting of maximal group size, time constraints, and group rules. Each vehicle evaluates the received messages depending on its relevance. It joins the group, if the goal or partial route fits. Individual actions for vehicles are to speed up, slow down and communicate with other vehicles in order to avoid conflicting situations of slow or blocked vehicles in front; also, vehicles may join for coordinated actions. Group members can choose lanes and speeds and act dynamically, thus improving their travel times and the traffic flow. One benefit of group formation is simplified control. Decisions are prepared by group leader and other joining group vehicles contribute to group goals, performing local optimization. Within a group, information sharing is technically possible between group members and opens new coordination and knowledge horizons.

6.1.5 Realistic Southern Part of Hanover Scenario

In Figure 6.7, the southern part of Hanover, Germany is illustrated - a realistic traffic network with two parallel and five perpendicular streets. The scenario was modeled and simulated on two main traffic simulators (see the reasoning in Chapter 4). One is based on the commercial microscopic traffic simulator AIMSUN, pictured in Figure 6.5, which has been significantly extended to support multi-agent models and vehicular communication. Then, in order to solve the problem of the shortage of multi-agent models and vehicular communication, the AIMSUN traffic simulation was combined with JADE, which enables agent-design and communication features. The same scenario was implemented with the open-source microscopic traffic simulator SUMO, illustrated in Figure 6.6, which has an interface for extension with multi-agent models and is also extendable with vehicular communication.

While AIMSUN and SUMO feature precise modeling of single vehicles, traffic lights and urban road infrastructure, the simulation of vehicular communication and individual decision making is not supported readily. Therefore, AIMSUN is connected with JADE agents and SUMO with Jason agents, both of which support communication features.

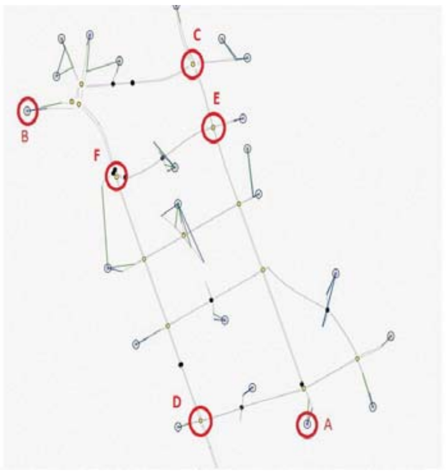


Figure 6.5: Representation in AIMSUN used for the PLANETS project.

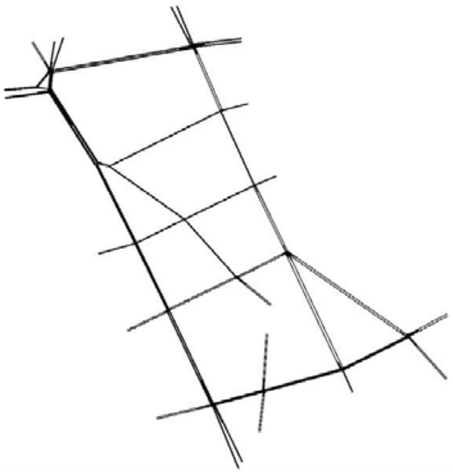


Figure 6.6: Representation in SUMO used for this thesis.

Figure 6.7: Southern Part of Hanover, Germany.

This realistic network scenario demonstrates the routing of vehicle groups in the combination of centralized control with decentralized decisions made by the traffic participants. The scenario is divided into three equal parts for the phases or like the green wave scenario - this needs simulation to see what is better.

The realistic Hanover scenario is modeled in AIMSUN and SUMO for research investigations and used in several own or peer publications by `Fiosins` [104] [105] [106] [111] [107] [110] [112] [109] [110] and `Görmer` [139] [141] [137] [138] [140].

6.1.6 Artificial Manhattan Grid Scenario

Whereas the aforementioned scenario 6.1.5 is not too different from a grid, Manhattan's street grid is a common urban street layout and easy to model. In contrast to the spoke-and-wheel layouts of cities such as London or Paris, Manhattan's rectangular street grid is planned, which places streets and avenues along mostly horizontal and vertical directions. With a length of over 10000 km, New York has one of the longest street networks of the world. New growth needed to be integrated and has formed a geometrically simple yet complex urban structure. The grid structure increases the island's dense, mixed use:it is walkable and transit friendly. Mobile Adhoc Networks (MANET) and other research is fascinated by the grid road topology. For this thesis, a model of 10 intersections with traffic lights and 100 meter lanes in between them was created. The phases were pragmatically designed bottom up, starting with the pre-phase in the South before the first traffic light appears, the main phase between the traffic lights, and the post-phase in the North, leaving the network. Here the scenario is divided into three equal parts for the phases.

For generating such a grid net, SUMO provides an internal tool called 'net-generate', which offers all the necessary options. This tool sets the amount of

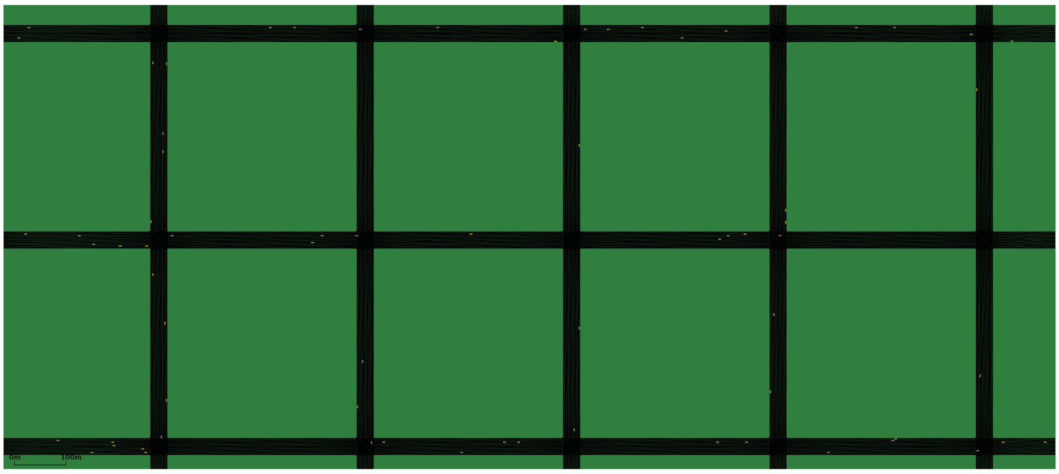


Figure 6.8: Zoomed in View of Manhattan-like Grid modeled with SUMO.

nodes and lengths of the streets. Additionally, parameters like the amount of lanes and the maximal speed can be determined, for non-defined values, automatic default values are used. For using the console, the example of the grid network with 10 nodes and edges of 100 meters is shown as an extraction in Figure 6.8 [2]:

```
Netgenerate.exe  g  grid.number=10  grid.length=100.0
```

The following listing shows additional values of the maximum speed and the amount of lanes. :

```
Netgenerate.exe  g  grid.number=10  grid.length=100.0
 default.lanenumber=2  default.speed=50.0
```

For avoiding turnaround, the following statement needs to be added:

```
--no-turnarounds
```

The name of the output file is given in the following:

```
-o=name.net.xml
```

Trips for vehicles are generated in two steps, 'randomTrips.py' and duarouter, as also presented in the following Section 6.2. The first is a Python script which can create different trips based on the net file. By default they are randomized, but can be influenced during initialization. After creating the trips, the

[2]Further information and options at: `http://sumo-sim.org/userdoc/NETGENERATE.html`

duarouter generates a name.rou.xml file for generating routes and assigns their departure times [3].

Adding self-defined vehicle types is a new but not yet mature feature in 2014. The output of randomTrips.py is faulty. Thus, a self-created script was designed, which is presented in the AppendixC.2.1.

6.2 Environment: Traffic Simulation

The open-source traffic simulator SUMO is used for the environment and includes various applications which perform traffic simulation in street networks. In 2005 [214] and still up to now, the following modules exist[4]:

- a network conversion utility which capable of importing networks from other simulation packages such as VISUM, VISSIM or ARTEMIS and from ArcView databases as well as from native XML descriptions,
- an OD-matrix to trip converter,
- a router required for traffic assignment and
- two different versions of the simulation: one completely without a visualization, used to perform fast simulations in a loop, and a second one, slower, but with a graphical user interface written using openGL, which is nevertheless relatively fast.

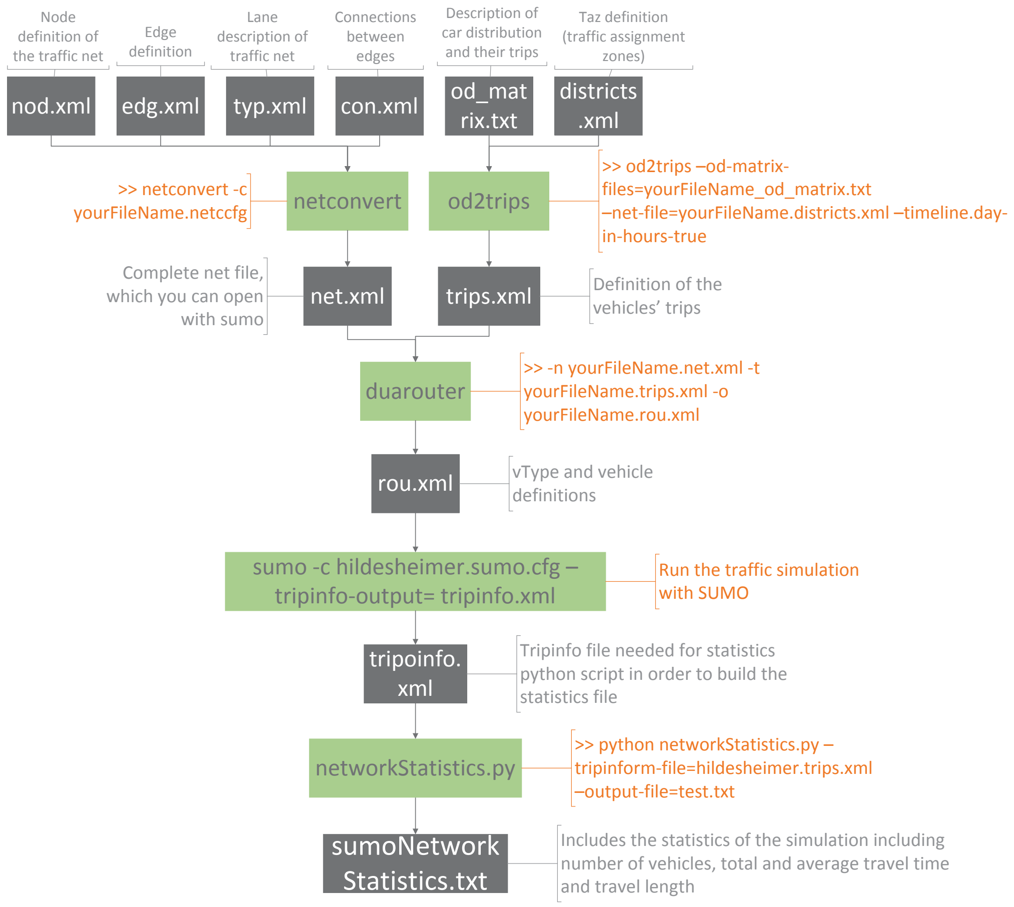
6.2.1 Creating Environment

When creating the environment with the open-source SUMO traffic simulator, the following process of creating a traffic net with OD demand and statistics file is captured in Figure 6.9, which needs to be designed. SUMO includes a statistics tool which can be used for travel times and other trip data of the vehicles.

First (as illustrated in Figure 6.9 at the top), the traffic net needs to be created with node and edge definitions, lane descriptions as the type and connections between the edges. This is put into 'netconvert' to complete a net file which SUMO can open. At the same time, also the description of the car distribution and their trips in form of OD matrices and traffic assignment zones is put into 'od2trips' to create the definition of vehicles' trips in a trips file. Then, in the upper middle of Figure 6.9, the 'duarouter' combines the net and trips files and creates the route (rou.xml) file with vehicle types and definitions. On the base of the route file, the traffic simulation with SUMO can run and 'tripinfo' can be one possible additional output. For 'tripinfo' a python script (see Appendix C.2.1) is needed in order to create the statistics file 'network-Statistics.py' as an output. Lastly, a text file 'sumoNetworkStatistics.txt' is the

[3]Note: `http://sourceforge.net/p/sumo/mailman/message/32364101/`

[4]All SUMO modules can be seen here: `http://sumo.dlr.de/wiki/Developer/Implementation_Notes/Sumo_Modules`.


Node definition of the traffic net
Edge definition
Lane description of traffic net
Connections between edges
Description of car distribution and their trips
Taz definition (traffic assignment zones)
nod.xml
edg.xml
typ.xml
con.xml
od_matrix.txt
districts.xml
netconvert
od2trips
Complete net file, which you can open with sumo
net.xml
trips.xml
Definition of the vehicles' trips
duarouter
rou.xml
vType and vehicle definitions
Run the traffic simulation with SUMO
tripoinfo.xml
Tripinfo file needed for statistics python script in order to build the statistics file
sumoNetworkStatistics.txt
Includes the statistics of the simulation including number of vehicles, total and average travel time and travel length

```
>> netconvert -c yourFileName.netccfg
>> od2trips –od-matrix-files=yourFileName_od_matrix.txt –net-file=yourFileName.districts.xml –timeline.day-in-hours-true
>> -n yourFileName.net.xml -t yourFileName.trips.xml -o yourFileName.rou.xml
sumo -c hildesheimer.sumo.cfg –tripinfo-output= tripinfo.xml
networkStatistics.py
>> python networkStatistics.py –tripinform-file=hildesheimer.trips.xml –output-file=test.txt
```

Figure 6.9: Overall flow of creating traffic net file with OD-matrix and statistics file.

output, including the statistics of the simulation with the number of vehicles, total and average travel times and travel distance.

Configuring and creating the environment in SUMO is described in detail in the Appendix C.2.1.

6.2.2 Initializing of Vehicle Groups

Here the sketching of vehicle groups is demonstrated:

- provide a traffic network environment i.e., in a traffic simulation.
- give each vehicle its own coordinate system (initial digital map) with an individual route
- each vehicle senses its relative position and orientation to others (requirement for group formation). Each vehicle has a relative feedback position-that is how vehicles are stabilized in computer simulations (compare to the paper by Yamaguchi et al. [384]: Formation is controllable by vectors with symmetrical robots and proven mathematically by a two-dimensional 'formation vector'.)

- equip each vehicle with an autonomous on-board unit allowing communication, which is essential for group formation, also called in-vehicle systems of automated vehicles, cooperative systems of connected vehicles and platooning ability, for example with dedicated lanes.
- In purely decentralized systems no superior exists, which can be modeled with the agent paradigm, but is not feasible in realistic traffic. Groups aggregate joint individual interests and can also be addressed by the infrastructure in a hierarchical traffic management, where there are, from the top down, the infrastructure, the vehicle-to-vehicle cooperative groups, and the individual vehicle level.
- When grouped, set the standard gap to a minimum group gap as illustrated in Figure 6.10. Due to the work on car following models, two values are used for vehicle length. The length-attribute in SUMO describes the length of the vehicle itself. Additionally, the $minGap-attribute$ describes the distance to the leading vehicle. Within the simulation, each vehicle needs - when ignoring the safe gap - $length + minGap$. But only the length of a vehicle is marked on the road as being occupied. The details of microscopic traffic simulation are addressed in Chapter 2.2.

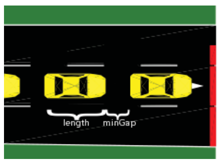


Figure 6.10: Minimal Gap for Vehicle Groups.

6.3 Agents: The Design of Autonomous Vehicles

Agents, in this case the vehicles, need to be reactive to the environment (for instance, if the traffic light turns red, the vehicles need to decelerate and finally stop), situated (present in the traffic network) and to have a certain degree of autonomy (i.e., their route choice). The individual vehicle is benevolent and cooperative without selfish or malicious behavior.

The following actions can be executed by agents:

- Agent actions for communication are *send* and *receive* with the following decision in response to the information
  - Action: recommendation, decision, move, lane change, etc.
  - Store: keep in memory
  - Discard: non-relevant information

- Agents can also react to the environment with movements to ***go*** *faster/speed up*, to ***stop***, *go slower/slow down* or to ***turn*** *left or right*.
- Actions regarding grouping involve ***advice*** *to join a group, accept, reject, no answer, confirmation*, or **negotiating**, **interacting**, **decisions**, and **results**.

A possible architecture of a vehicle agent is shown in Figure 4.7 (cf. adapted from our publication [110] p. 73.). A simulated vehicle is equipped with the following modules, called Apps:

- Learning App: is an adoption of the weights provided by the TMC and their combination with the individual weights
- Routing App: provides access to the routing services
- Grouping App: contributes group formation and corresponding decisions
- CommBox: is responsible for the communication module (V2I and V2V)
- setRoute App: is a translation of a route to corresponding actions (turns)

6.3.1 Homogeneous Vehicles

The homogeneous type of autonomous vehicle is the default vehicle provided by SUMO[5]. It consists of three parts: the individual vehicle, the physical properties, and a route. Several vehicles can share the same route or vehicle type. The definition of a standard vehicle type according to `Stefan Krauß'` thesis [216] [6] is:

```
<routes>
    <vType id=”type1” accel=”0.8” decel=”4.5” sigma=”0.5”
    length=”5” maxSpeed=”70” color=”1,1,0”/>
<routes>
```

This initial definition describes the physical parameters of a vehicle, such as its length with 5 Meter as its net-length, its color yellow, or its maximum velocity of 70 (in m/s). The initial Krauß car-following model's parameters are not the maximum acceleration possibilities of the vehicle but rather what the driver chooses (imperfection of the driver is measured by the sigma between 0 and 1) in combination with the physical possibilities of the individual vehicle (i.e., like a fast Porsche) and the environmental context (it is not advisable to accelerate to 100 km/h in 5s).

The vehicle type is defined in SUMO with attributes starting with small letters.

```
<vType   id="normalCar"
accel="4.0"      decel="4.0"  length="6.0"    minGap="5.0"
maxSpeed="10.0" speedFactor="1.3"
tau="1.5"        sigma="0.5"  color="137,137,137"/>
```

[5]available on the website sumo-sim.org

[6]`http://www.sumo.dlr.de/userdoc/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html`

Attributes are:

- *id*: unique id of the vehicle type
- *accel*: acceleration of the vehicle
- *decel*: braking power of the vehicle
- *length*: how long the vehicle is from front to end
- *minGap*: minimal distance to the front vehicle
- *maxSpeed*: the maximum speed the vehicle can drive
- *speedFactor*: in reality, every driver has their own opinion about speed limits. With this factor, the driver will drive 1.3 times as fast as the current speed limit of the edge.
- *tau*:not every driver has a fast reaction. tau defines a delay in seconds. Only positive numbers are allowed!
- *sigma*: the driver's imperfection. How well/how closely does the driver stick to the parameters? (0-1)
- *color*: rgb color definition

SUMO itself has defined some default values for vehicles[7].

6.3.2 Heterogeneous Vehicles

It is a cumbersome task to represent all different types of vehicles: small cars, compact class, luxury cars, sports cars, convertible cars, vans, electric cars, middle class, upper class, vintage cars, super sports cars, muscle cars, sport utility vehicles (SUV) and cross country vehicles. Due to the aim of grouping vehicles, the interesting characteristics are the vehicle length, acceleration and deceleration properties, and maximum speed. Vehicle length is interesting because it determines how many vehicles can form a group. If they are longer, then less vehicles fit on a section before a traffic light whereas, if they are small, more fit. Here, the three lengths small, regular (standard and sporty) and long are defined. There is a distinction between the maximum speed and acceleration of a vehicle because catching up with others might influence joining a group. That is why the multitude of vehicle types is set in four categories of vehicles for the simulation of vehicle groups.

The vehicle types represent the four main classes of vehicles in typical Western city traffic (1) the sporty fast car, (2) the long truck or bus, (3) the small vehicle, and (4) the regular vehicles.

For realistic data, the properties of the four vehicle types are based on the exemplary vehicle models of the four categories:

- slow-vehicle: SMART[8] -high speed: 145 km/h acceleration: 16,8s in 0-100km/h length: 2700 mm

[7]They can be found here: `http://www.sumo.dlr.de/userdoc/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html`

[8] http://www.smart.de/de/de/index/smart-fortwo/pure.html

Table 6.1: Characteristics of heterogeneous vehicles.

vehicle type	**Acc/Dec**	**length**	**max speed**
(1) Aston Martin	6.04 / 6.0	4.71	81.9
(2) MAN (Mercedes Bus)	1.5 / 4.5	11.98	25.0
(3) Smart	1.65 / 4.5	3	40.28
(4) Golf GTI	4.27 / 5.0	4.27	68.33

- fast-vehicle: Aston Martin DB9[9] -high speed : 295km/h acceleration: 4,6s in 0-100km/h length: 4710 mm
- normal-vehicle: VW Golf GTI[10] -high speed: 246 km/h acceleration: 6,5s in 0-100km/h length: 4268 mm
- long-vehicle: MAN or Mercedes Citaro LE City Bus[11] -accel: 0.4 [12] decel: 0.9 length: 12170 mm for a standard solo bus speed: 220KW (300 PS)

The Aston Martin stands for a fast sporty car, which can also include Porsche, Ferrari etc. with a normal length. Whereas the MAN (or Mercedes City Bus) stands for a long vehicle, usually a truck or bus with carefully calculated acceleration and deceleration - not because they do not have the power, but due to their fragile load of goods or people. The Smart represents the class of small city vehicles like bikes or small cars like the Mini and usually does not have a high maximum speed, but good characteristics for city traffic. The Golf (GTI) represents the class of normal standard vehicles - which is also used in the homogeneous vehicle characteristics.

To map the data to the simulation platform SUMO, the following calculation formulas were used:

- high Speed : $(xkm/h * 1000)/3600s$ result is in m/s; x high speed of the model in reality km/h
- accel: $\frac{(100000m/3600s)}{time}$ //time = time to get from 0 to 100 km/h; result is in m/s
- minGap = $\frac{highSpeed}{2} + x$ //x = estimated value. Example: Bus has low high speed, but a lot of weight, therefore add an additional 10 m for safety (personal opinion) //Driving school teaches minimum gap should be at least half of the current speed.
- decel = 6-8 for cars[13]; Trucks/Buses less (Bus because of passengers/ truck because of trailer and cargo, which pushes it forward)

The following are the example values used for the vehicle properties:

[9]http://www.astonmartin.com/de/cars/the-new-db9/db9-carbon-edition
[10] http://www.volkswagen.de/de/models/golf-gti.html
[11] `http://www.mercedes-benz.de` as an alternative MAN Bus due to available values `http://www.bus.man.eu/global/de/stadtbusse/man-lions-city-le/technik/motor-und-getriebe/Motor-und-Getriebe.html`; length 11980mm, no data for speed(250 PS), no values for accel and decel.
[12]accel and decel values from a contact at EvoBus
[13]`https://de.wikipedia.org/wiki/Bremsweg`

Table 6.2: Distribution of heterogeneous vehicles.

vehicle type	**real statistic**	**simulation distribution**	**amount=5275:4**
(1) Aston Martin	1%	5%	263
(2) MAN	6%	20%	1055
(3) Smart	92%	25%	1318
(4) Golf GTI	1%	50%	2637

```
    Acceleration: 100km/h = 27.778m/s [time]
    minGap = highSpeed/2 + x //x = estimated value.
    Example: a Bus has a low high speed due to a lot of weight or passengers
    => maybe add additional 10m for safety
    decel = 6-8 f r car (estimated)
```

Routes are also defined using the vehicle types and are inserted as follows:

```
<routes>
 <vType accel="6.04" color="1,0,0" decel="6.0"
 id="astonmartin" length="4.71" maxSpeed="81.9"
 minGap="2.5" sigma="0.5"/>

 <vType accel="1.65" color="0,1,0" decel="4.5"
 guiShape="passenger/hatchback" id="smart" length="3"
 maxSpeed="40.28" minGap="2.5" sigma="0.5"/>

 <vType accel="4.27" color="0,0,1" decel="5.0" id="golf"
 length="4.27" maxSpeed="68.33" minGap="2.5" sigma="0.5"/>

 <vType accel="1.5" color="1,1,0" decel="4.5"
 guiShape="truck/trailer’" id="man" length="11.98"
 maxSpeed="25.0" minGap="2.5" sigma="0.5"/>
</routes>
```

The distribution of heterogeneous vehicles is set according to the schema in Table 6.2. In the simulation, typical urban German distribution of vehicles is used:. 50% normal standard vehicles, 25% small vehicles, 20% long vehicles, and 5% fast vehicles. Because this research investigates whether vehicle groups improve rush-hour traffic in cities, the amount of vehicles in a simulation need to be adjusted to the level of service - category C or D. This is done by sending a fixed density of vehicles into the simulation with the heuristic value that a maximum of 5275 agents represents the vehicles.

The calculation for Table 6.2 can be broken down into:
distribution for 100 vehicles:

- smart: 1/4
- golf: 2/4
- aston martin: 1/20
- man: 1/5
- distribution for 5275 vehicles / 4 vehicle types:
- 5275 * (1/4)
- 5275 * (2/4)

- 5275 * (1/20)
- 5275 * (1/5)

which means, to reach the amounts for a total of 5275 agents in one simulation:

- 5275:2 is 2637,5 equates to 50%
- 5% of 5275 is 263,75
- 20% of 5275 is 1055
- 2637:2 is 1318,5 equates to 25%

The real statistic in Table 6.2 is taken from the Kraftfahrtbundesamt (Department of Motor Vehicles) of Germany to find the distribution of the vehicles[14] and taken into account.

The **vehicle settings** can be found under: project/MATI/lib/SUMO/tools script.

6.3.3 Agent Implementations

The behavior of MATI agents created in the Jason interpreter changes with the way they are implemented, as shown in the following. MATI agents are created with JAVA and are empty at the beginning - just like the grouping_agent.asl.

Interpreter File

In each scenario the interpreter file, where the simulation function is defined, is set globally .

```
{
   "simFunc" :
      {
         "classString" : "mj_func.CAEDF"
      },
   "artifacts" :
      [
       {
         "classString" :
         "mati.interpreters.grouping.CGroupingArtifact"
       }
      ]
}
```

Listing 6.1: Interpreter File.

Here it is the **m**ati-**j**ason function with the **C**lass **A**dopted **E**uclidean **D**istance **F**unction and the artifact to the grouping artifact. Another option of the simFunc would be the extended Manhattan Distance Function.

Zigzag Agents

For example, there are multiple ways to implement a Zigzag Agent.

[14] `http://www.kba.de/DE/Statistik/Fahrzeuge/Bestand/bestand_node.html`

Agent 1:

```
// goal is to zigzag between lanes 100 times
!zigZag(1000).

// abort criterion
+!zigZag(N) : N == 0.

+!zigZag(N) : N > 0 & ((N mod 2) == 0)
   <-   changeLane(left);
        !zigZag(N-1).

+!zigZag(N) : N > 0 & ((N mod 2) == 1)
    <-  changeLane(right);
        !zigZag(N-1).
```

Listing 6.2: ZigZag Agent 1.

Agent 2:

```
// goal is to zigzag between lanes 100 times
!zigZag(1000).

// abort criterion
+!zigZag(N) : N == 0.

+!zigZag(N) : N > 0 & ((N mod 2) == 0)
   <-   changeLane(left);
        !!zigZag(N-1);
        .print("This line is executed right away")
        .

+!zigZag(N) : N > 0 & ((N mod 2) == 1)
    <-  changeLane(right);
        !!zigZag(N-1);
        .
```

Listing 6.3: ZigZag Agent 2.

Note the difference in how the Zigzag plan is called in each agent. The first agent uses a single exclamation mark, which means the plan is called recursively. The second agent uses a double exclamation mark, which means the code is executed right away after the call. It spawns a new thread for the execution of the plan. The result is that Agent 1 will run faster than Agent 2, because it is recursive, which is more efficient in the AgentSpeak Language.

The following example from the Jason book [49] is used as a base:

```
!print_fact(5).

+!print_fact(N)
    <- !fact(N,F);
        .print("Factorial of ", N, " is ", F).

+!fact(N,1) : N == 0.

+!fact(N,F) : N > 0
    <- !fact(N-1, F1);
        F = F1 * N.
```

Listing 6.4: Jason Example.

The final Zigzag agent is implemented with the Artifact concept introduced by [173] and uses the JaCaMo interpreter using Jason combined with the environment CarthAgo and the Moise organizational concept:

```
perceptArtifact("vehicle",
"mati.interpreters.jacamo.artifacts.VehicleArtifact").
opposite(1,0).
opposite(0,1).

!start.

+!start
    <-
    log("Changing to left");
    !changing(1);
    .

+!changing(Side)
    :opposite(Side,OtherSide)
  <-
    log(Side);
      changeLane(Side);
      !!changing(OtherSide);
    .

+!changing(Side)
  <-
  log("alternativfall");
  .

-!changing(Side)[error_msg(Msg)]
    <-
    log("error: ", Msg);
    .
```

Listing 6.5: Final ZigZag Agent.

ColorSort Agent

Agent Speak Language (asl) agents can also be written, such as the colorSortAgent.asl, which is used for the preliminary phase in centralized group formation:

```
+!changeLane(X)[source(S)]
    : (S == self | S == simpleColorSortArtifact) &
      laneIndex(Index) &  X > Index
    <-
        changeLaneByIndex(Index+1);
        mj_ia.info("I should change to lane ",X,".
        Change Lane!!!");
        !changeLane(X);
    .
+!changeLane(X)[source(S)]
    : (S == self | S == simpleColorSortArtifact) &
     laneIndex(Index) &  X < Index
    <-
        changeLaneByIndex(Index-1);
        mj_ia.info("I should change to lane ",X,".
        Change Lane!!!");
        !changeLane(X);
    .
+!changeLane(X)[source(S)]
    : (S == self | S == simpleColorSortArtifact)
    <-
        mj_ia.info("I am on the ",X,". Lane!!!");
    .
```

Listing 6.6: Color Sort Agent.

6.4 Interaction: Communication between Traffic Participants

Vehicle cooperation based on multi-agent group models was studied with mechanisms for V2V communication. Without interaction there would be no groups. Since communication is still under ongoing research and influences time and performance parameters, this thesis tries to avoid communication to the extent that group formation is still feasible, but communication is minimized (agents would like to send everything, but for vehicular communication protocols, frequencies and regulations need to be met). In Multi-agent Systems (MAS), agents can interact through communication or by reacting to changes in the environment or to the behavior of other agents. Both interaction methods are discussed.

The first approach is used in AIMSUN and JADE, where the environment is represented through communication, then later with Jason and, therefore, MATI and JaCaMo, in which the environment is perceived by the view range of an agent and calculated based on its internal plan for the next action [253]. However, for communication, agents must be able to understand each other, which can be achieved by using a common language. In the JADE approach, FIPA ACL is used, because it is a scientifically recognized standard. FIPA ACL provides ontologies on the basis of vocabularies for a common language. In JADE, the appropriate ontology was implemented in the environment in Repast S, so that each agent connected to the environment 'speaks' the same language. The AIMSUN approach uses wireless LANs, described below, and Jason uses the AgentSpeakLanguage for communication, addressed in Subsection 6.4.1.

In contrast to the desired macro effects in models and simulation that result from the system level interaction of individuals, the individual agents behavior, their communication and organizational structure needs to be specified by designers at the micro-level.

Most of the knowledge about realistic communication in urban networks arose from talking with colleagues from the PLANETS project (using the AIMSUN simulator), especially with `Hugues Tchouankem`[338] [339][15], on which we worked together and which was published in `Görmer et al.` [137] and `Fiosins et al.` See [110] on multi-agent systems and communication in traffic networks, which this Section is based on.Figure 6.11 shows communication ranges with shadowing effects on two nodes, representing the intersections Aegidientorplatz and Marienplatz in Hanover, Germany.

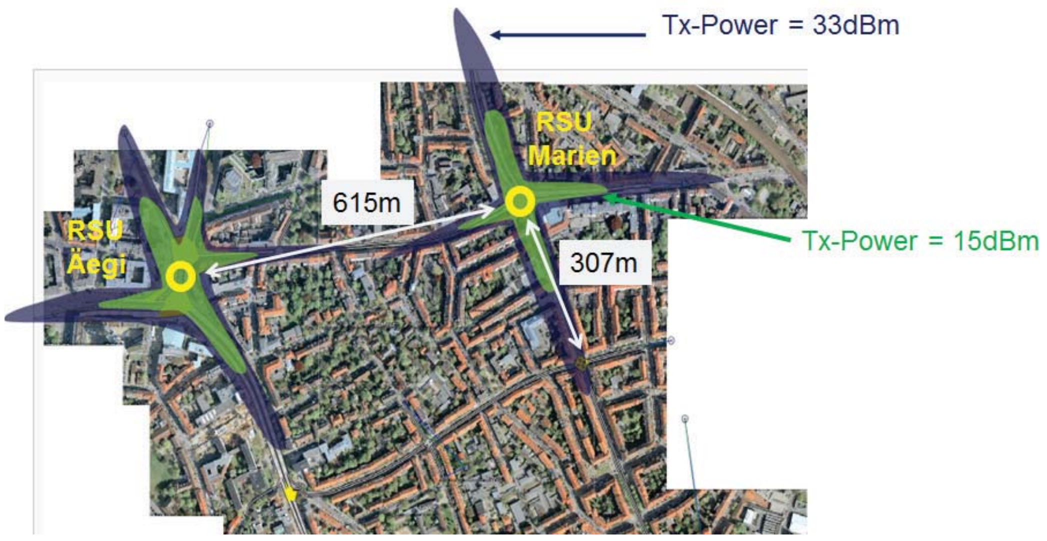


Figure 6.11: Communication Range with Different Frequencies of Road Side Units (RSU) at Aegidientorplatz and Marienplatz, Hanover, Germany.

[15]Institute of Communications Technology, Leibniz Universität Hannover, Hanover, Germany.

In the future, safety and comfort will increase through wireless communication for the driver. Infrastructure networks (Vehicle-to-Infrastructure, V2I) and Ad-Hoc networks (Vehicle-to-Vehicle, V2V) are the new technologies for wireless communication between the vehicles and their environment. Combined network topologies are called Vehicle-to-Xchange (V2X) communication and are the subject of development projects in different research works, which try to define a standard. In traffic management more dynamic decentralized approaches are possible due to progress in the technology of sensors and communication. Thus, `Fiosins et al.` [108] p. 5 states:

> The goal is to extend existing traffic management with extensions for more dynamic components on the base of V2I communication.

In traffic, those network topologies are very dynamic in contrast to conventional wireless LANs, and therefore new models are needed and vehicles need communication systems.

Usually, Multi-agent systems (MAS) assume perfect information exchange between agents and have independent models for interaction with the FIPA ACL standard without real-world communication technology. But, with the addition of the realistic IEEE 802.11 wireless LAN, the reliability and capacity of communication have to be considered and dedicated models for communication are required.

For the research question of decentralized group formation, the models and methods of communication are essential (compare [108] p. 5). Group models and their information exchange depend on communication performance. Also, in MAS, the real-time information exchange between agents is addressed in order to guarantee a reliable interaction of traffic participants. For realistic simulation of MAS, a communication model including a complete communication protocol stack and, especially, a radio propagation channel, are important design decisions. Urban environments need to take into account the frequency-selective and time-variant character of the radio channel due to buildings causing shadowing, reflections and scattering as illustrated in Figure 6.11.

A combined figurative example of multi-agent group formation with the use of communication is seen in Figure 6.12, including the impact of the radio shadowing model. The vehicular group formation is indicated by the dark blue area with a green colored leader (first vehicle in the left lane) and red colored group members heading from south to north, as indicated by the arrow. Black vehicles are not in any group and the orange vehicle is analyzing and comparing the group plan with its own in order to pass the next green phase. There is also another group with two vehicles and one non-grouped black vehicle on the right hand side of the picture and another orange vehicle on the left, waiting at a traffic light, receiving a CAM and analyzing the group plan in comparison with its own. The dashed circle around vehicle A marks the complete communication range, which is partially blocked by surrounding buildings.

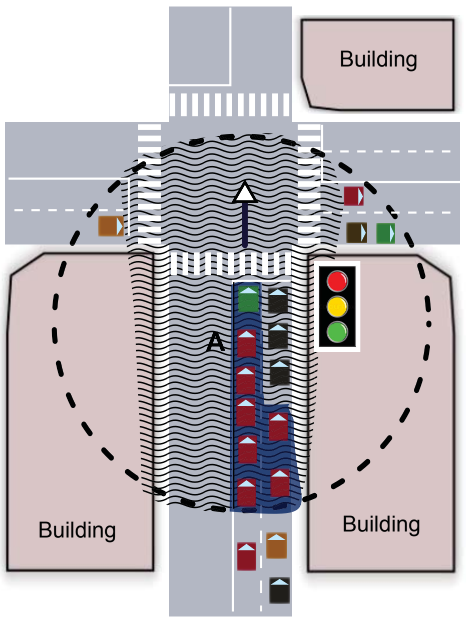


Figure 6.12: Combined example intersection with multi-agent group formation and the use of communication including the impact of the radio shadowing model [from [110] p. 80].

6.4.1 Jason

Jason also finds recognition in the Background Chapter 2 in Section 2.3.5 and Chapter 4 in Section B.4. Additionally, the use of Jason in MATI is described in 4.3.5. Therefore, Jason is mentioned here, but only JaCaMo is described in detail to avoid redundancies. Jason uses the AgentSpeakLanguage for communication.

6.4.2 MATI

The MATI agents communicate with the standard communication function of Jason. The use of JADE agents designed according to the FIPA ACL communication standard is possible, but for MATI agents there is no added value compared to the communication standard in Jason. This means thatJADE can be used and combined, but does not give an additional communication benefit for the purpose of vehicle groups.

The MATI implementations have already been described in Subsection 6.3.3.

6.5 Organization: Vehicle Groups in Cities

This thesis tackles grouping autonomous vehicles for urban traffic. Public transport already groups passengers into vehicles. But passengers, drivers and human-related conditions are not taken into consideration for this research. Automated highway system (AHS) [38] coupled eight to twenty-five normal vehicles into platoons either electronically or mechanically (to road trains), decreasing the distances between them, but needed infrastructure changes, namely, magnetized stainless steel spikes for lane location. Urban networks often do not have long streets, but lots of action choices and influencing infrastructure like traffic lights and signs. Urban autonomous groups are expected to be smaller, in the range of three to twenty vehicles, compared to highway platoons. They are called groups instead of platoons. Infrastructure incentives for groups or group sections like a variable message sign indicating that grouping is desired or a guarantee for vehicle groups for green phase passing would be advantageous, but are not considered in these investigations.

In this section, the design of vehicles for decentralized and centralized organization and cooperation is described, such that autonomous vehicles can form groups and reach their goal more efficiently than when driving individually. The organization structure of urban traffic groups and its algorithms are outlined in the following.

The process of group formation requires coordination and time effort, so where and when it is triggered needs to be well thought through because of real-time traffic conditions. There are many group formation methods presented in the model in Section 5.4, but not all are easily applied to realistic traffic. Deductions need to be made for vehicle groups in favor of the interplay of different domain combinations of traffic management, agents and communication. Each application field has its own requirements and assumptions which do not fit together so reasonable compromises need to be made. For example, agent theory usually assumes perfect and complete (without package losses) information and communication, whereas traffic management does not want to distribute all information (due to movements toward more dynamic traffic plans), i.e., internal signal plans (there are traffic independent and traffic dependent signal plans), just the outcome in form of red, yellow and green lights. Communication deals with sending frequencies, radio channels, shadowing effects and package losses or multiple sending because of limited communication ranges, which already indicates that perfect and complete communication is very ideal and not realistic.

Possible grouping strategies for vehicle agents to conduct tasks together for a common purpose are quite limited in a world where individual driving is very prominent. By functional requirements the traffic network and the individual vehicle need to profit by reducing travel and stop times and, thus, increase the traffic throughput, which is especially relevant in rush-hour traffic. Therefore, dense traffic in peak hours as well as grouping sections needs to be identified. During the night traffic or rush-hour complete congestion, grouping vehicles with speed and gap adjustments cannot provide relief due to limited group actions. Anyhow even in light or congested traffic grouping can provide possible improvements through communication and coordination.

Grouping input and output requirements are identified with their route choice from origin to destination and their interdependent sub-plans, sorted by the same or common goal of same route, green phase passing and next steps and decisions. Traffic objects of light signals and management messages as well as other moving vehicles are taken into account and may cause re-routing or re-grouping.

In order to investigate group effects and stability, simulations need to be executed as default behavior (without vehicle groups as the default simulation), with general vehicle groups of the same kind presented, in Subsection 6.3.1, and with a differentiation of four vehicle types having different characteristics i.e., normal, slow, fast, and long, as described in Subsection 6.3.2.

Autonomous vehicles are implemented with the design of agents as modified software programs in traffic simulators, as discussed in Section 6.3. Traffic participants can be divided into mobile vehicles and mostly static infrastructure objects. Both vehicles and infrastructure objects can be grouped, but the focus of this thesis is the grouping of vehicles. There are individual vehicles before and after being grouped and autonomous vehicle groups, which have their strategies. This could imply group priority by the infrastructure, as is already done with public transport buses or trams, and new concepts of rules, norms and regulations for groups need to be created. Nowadays, infrastructure elements are mostly centralized, but new technologies to make traffic lights more traffic-dependent are applied in dynamic traffic management. Infrastructure decisions could also be automatized with the agent paradigm and thus, new methods tried for more decentralization i.e., communication, but still make the overall system controllable.

The work for this thesis has undergone different approaches with traffic simulators and agent interpreters, as discussed in Chapter 4, and, accordingly, three lines of vehicle groups have been developed and implemented (differences were discussed in Chapter 4; up to this end they are not compared due to many technical differences in implementation):

1. for the PLANETS project with AIMSUN in the programming language C++,
2. for ATSim, combining the traffic simulator AIMSUN (C++) with the agent interpreter JADE (JAVA) and
3. for MATI using the open-source traffic simulator SUMO (C++) with TraCi4J interface and the agent interpreter Jason (JAVA).

The general organization structure is discussed above. This thesis aims for a very decentralized approach on autonomous vehicle groups with the help of emerging cooperative technologies, while respecting the present hierarchical infrastructure. For simplification in the very complex domain of autonomous traffic, the implementation of centralized vehicle groups, which generally are compulsory and static, is discussed as well as the decentralized vehicle groups, which act and react in a real-time environment.

This thesis makes a distinction between three classes of groups, i.e., centralized static, decentralized hierarchical and mixed dynamic groups.

Static groups can be recognized by the classical static control, which usually uses a central architecture. They result from a traffic management center, a central control which gathers all traffic data in order to make expert decisions including all the domain constraints, e.g.,

1. ensure traffic flow throughout the network;
2. give preference to main highways;
3. optimize signal plans according to historic data;
4. ensure traffic safety;
5. optimize traffic infrastructure according to growing traffic demand.

Other group formation methods are discussed in the following Subsections with the evolution of different group formation approaches for the projects

- AIMSUN - centralized vehicle groups,
- ATSim - decentralized hierarchical groups and
- MATI - mixed static compulsory and decentralized dynamic vehicle groups.

6.5.1 Centralized Vehicle Groups

The centralized vehicle groups are easier to implement in urban traffic nowadays due to experienced classical traffic management. Centralized groups are controllable and the effects of these urban platoons are predictable and therefore manageable by traffic management. Centralized vehicle groups need to be triggered by the infrastructure when traffic is dense and group formation can start at each traffic light with vehicles close in space and time. In the realistic scenario of the southern part of Hanover, Germany, the data was provided for the morning peak hour between 7:30 and 8:30 a.m.

For the mixed MATI static group formation, similar traffic flows were generated for the other scenarios of three lanes, Hildesheimer green wave and grid. Then the scenarios were divided into three sections for the preliminary, main and post phase to study the static grouping effects. The phases are static infrastructure elements usually designed at already existing traffic lights and do not stop the vehicle flow. That means that vehicles close to phase transition might not be sorted or grouped.

The preliminary phase sorts the vehicles depending on their route which is done with a color sort algorithm (see Appendix C.3.4) which needs space. The sorting can start with the first vehicles entering the lanes, which are closest to the end of the preliminary phase, or, in a filled network, sorting can start where the preliminary phase begins. The first type of sorting is better for unfilled nets and gives the sorting phase more time until the main group action phase starts; the second sorting is more realistic with a full net and continuous traffic flows.

The main phase uses a leader-member concept with the contract net algorithm (see Appendix C.3.5) to group the vehicles after the group leader of each lane and they continue driving in the group until the end of this phase.

The post phase is the adjourning of groups and returning to individual driving mode according to their initial route plan.

AIMSUN A variation of centralized groups has been done in the PLANETS project with AIMSUN. It describes a simple version of a contract net for the simulation (without specific phases): one group leader is appointed, namely, the first at a red traffic light, and all others check whether they have the same destination and use the group leader's plan to drive through the next intersections jointly. It is described in more detail in our ARTS conference paper [110] p. 79 (the Fig. 8 that is illustrated is redesigned for this thesis in Figure 6.12).

In the PLANETS project, the scenario of a realistic traffic network, shown in the three collected Figures in 2.16, of the southern part of Hanover, Germany, is considered. From the original map in Figure **??**, two parallel and five perpendicular streets are considered in the microscopic traffic simulations as defined in AIMSUN 6.5. The SUMO simulation of the same network is illustrated in Figure 6.6 and used for the MATI simulations.

In March 2009, real data was collected and used in the AIMSUN simulation with the goal to combine emerging technologies of data processing, autonomous agents and communication with the transforming traffic domain in order to be more dynamic and traffic-dependent.

In valuable discussions with the PLANETS colleagues, a good understanding of the different domains was established and compromises needed to be made for a good and realistic synergy between them. Basically, the decision was to 'see' or perceive the simulation world through communication, which was good for the general information sharing required by everyone, especially the data processing and autonomous agents. This way, the infrastructure elements of traffic lights communicated fixed signal plans (because they are more predictable) with the rest time value of the signal phase. Mobile traffic participants communicated when they entered and left the simulation and, upon request, where they are at a certain time, which is useful for finding vehicle neighbors which could join a group.

The simulation environment modeled in AIMSUN is an urban street network with mixed traffic densities. Units of density are passenger cars per lane per mile, or pcplpm. For this calculation, trucks and buses are converted to passenger-car equivalents. Average density values are determined for both directions and converted to level-of-service (LOS) ratings. The topology is a daytime dependent digital map.

The autonomous vehicles are equipped with a board computer, including navigation and communication, and know their destination. Vehicles plan their route through the network based on historical data. The vehicles are modeled as intelligent agents which can adapt to change. The vehicles can join groups if strategic or spacious goals are similar. The vehicles can concede their initial route for coordinated group behavior and adopt a route depending on the actual traffic situation.

Traffic lights can be regulated traffic-dependently in a separate simulation run, but for grouping they are traffic-independent. One goal was that traffic lights also be modeled as agents in order to coordinate themselves too, but this was not implemented. Traffic management gives recommendations to the traffic participants, static traffic lights and mobile vehicles. The communication infrastructure is joined with the traffic lights or Road Side Units (RSU) and communication is based on wireless communication protocols.

The goal from traffic management was coordinated direction change of vehicle groups with Vehicle-to-Infrastructure communication as well as reducing the traffic load. Traffic loads can be identified and partial goals can be determined with previous planning or dynamically. Traffic participants need to be cooperative. Vehicle-to-Vehicle infrastructure is used for group formation, which is triggered by recommendations for the Hildesheimer green wave street through traffic management. The vehicles' direction can be changed as a group with variable messages signs or individually via Vehicle-to-X communication.

From the agent perspective, the task is to describe a system of traffic participants with rational agents. Synergies can be found between each domain and:

- traffic management: The institutional norms of traffic agents' behavior in form of traffic rules and valid, legal and meaningful actions in different types of environments and topologies
- data mining and processing: Define cooperative behavior of agents as belief representation (knowledge about the system) and intelligent cooperative decision-making, for example, routing and grouping, conflict resolution, and learning (adaptive behavior)
- communication: Information exchange between agents including rules of the information exchange (when, whom and what to send), protocols of the information exchange (common language) and joining of beliefs (how to deal with received information)

The joint research question of traffic management and agents is to find the optimal level of cooperation between traffic participants and the optimal level of combination of decentralized vehicle behavior with centralized control. The general research question was the result of the penetration rate of added features (dynamic traffic dependent signal systems, wireless communication, routing with learning strategies and grouping), depending on the overall system performance. Therefore different experiments with different settings were conducted.

For the centralized grouping of vehicles in AIMSUN, all relevant classes are described in the following. From the complete scenario of the southern part of Hanover, Germany, Hildesheimer Street (described as a separate scenario in Section 6.1) was triggered by traffic management for grouping because it has a high traffic density (LOS C to E).

First, the 'vehicle class' with all its parameters is described. Then the 'Group' in which the vehicles are collected with their id and goal/destination, and 'GroupApp', which determines where and whether the grouping is active. The group leader has the special task of calculating the optimal group speed and then the realistic speed, which is described. This grouping concept is hierarchical; the group formation process is divided into two plans: one for the group leader and the other for the group members.

class Vehicle: when instantiating the class 'vehicle', all parameter are set. The function 'UpdateComSys' transmits to all the speed, position, sectionID, distance and the next section ID (see Appendix C.2.1 for more details).

Group: In the class 'Group' all vehicle participants are collected. The mobile vehicles are added with their personalized 'ID' and their goal (destination). All participants can be updated with the current speed, their position and the number of the lane where the group is located.

GroupApp: The class 'GroupApp' accesses the INI-file of AIMSUN where the strategy 'Grouping' is either set to active or not. 'GroupApp' contains a method 'setLeader' with which the leader is determined and set. First, the distance to the goal is determined. If the vehicle is first at the red traffic light, which is set as the grouping initialization, then this vehicle becomes the leader. This is described in more detail with a listing in the Appendix C.2.1.

The set 'Leader' sends all his information via a broadcast message to the rest of the vehicles in the possible group. The information contains his ID, his position, the street ID, the optimal speed and a leader flag. For the calculation of the optimal speed the method 'calculateOptimalSpeed' is called (see Appendix C.2.1).

ComSys: In the method 'manage' of the class 'ComSys' three different plans or strategies are located.

Plan 1 is called the group *leader* plan and described in the Listing D.5 in the Appendix D. Plan 1 starts with the decision 'tl_msg.empty != false' which indicates that there must be messages from the group leader or it stops. Then, the iterator message is filled with the first messages of the group leader 'message = tl_msg.begin'. The iterator checks if the end of message was reached with 'message != tl_msg.end' or it stops and checks if the vehicle is in the right section by 'isOnSection = true' or it stops and transmits the fitting signal phases. The next process transmits the fitting signal groups to the actual section 'appSG = getSGonSection'. The next decision regards the traffic light behavior state 0 = red, 1 = green, 2 = yellow, 3 = green blinking with 'sg_State = 1?' or it stops. In the following process the real optimal speed is calculated from the factors to the stopping line and the rest values of the green phase 'realOptSpeed = calculateRealOptimalSpeed' and 'curr_speed = GetCurrentSpeedoptSpeed = calculateOptimalSpeed'. The optimal speed is minimal if the optimum is smaller or maximal if the optimal speed is bigger in the last decision 'optSpeed >curr_Speed && optSpeed >realOptSpeed && curr_Speed >30.0'. This results in coloring the vehicle 'colorVehicle' and the optimal speed being transmitted to AIMSUN 'AKIVehTrackedModifySpeed'. Then the process restarts.

Plan 2 is named the group *member* plan, but also checks the leader preferences, since it depends on that plan. AIt is described in the Listing D.6 in the Appendix. Plan 2 starts and group messages 'gr_msg = getIncomingGroupingMessages' are collected. Then the group leader is set and transmitted 'setLeader IsLeader = getLeader'. This is followed by the decision 'isLeader = true?' and then a group message is generated and sent with a time stamp 'sendMessage'. The decision 'curr_sect_id = 42738?' is made because there is only one grouping traffic light where groups are created in that specific section entering the southern part of Hanover.

Then the color of the leader 'colorVehicle' is set and the optimal speed is sent to AIMSUN with 'AKIVehTrackedModifySpeed'.At this stage there are no group members 'count_group_member = 0'. Then the next Section Id and the destination are retrieved 'next_sect_id = GetIdSectionNext destination_centroid = getCentroidIdDest'.

The next decision 'gr.msg().empty = false?' is a message iterator of the group which is not allowed to be empty. Then the message iterator is filled with the first messages of the group 'message = tl_tl_msg.begin' and the iterator checks if the end has been reached 'messaged != tl_msg.end'. When the present vehicle is the leader 'IsLeader = true?', then the next section of leader is retrieved 'leader_next_section = section_id_next'.

After the next planned section the leader is to drive 'next_sect_id = leader_next_section?', the decision is also made as to which is the goal or destination centroid (the source or drain where vehicles are entering or leaving the simulation), 'destination_centroid = centroid_id_1?' or the other '2', '3' or '4'?. In the next process step, the distance to the destination is calculated with the plan 'distance2end = section_distance2end'. The remaining green time is set 'restgreen = rest_green', the distance to the leader is retrieved 'dist2leader = GetPosition', the maximum speed is set for the section 'max_section_speed = Get MaxSpeed' and the actual speed determined.

Then the grouping process 'isGroup = joinGroup' is performed if the parameters are the same (or similar) as the leader's by adding the vehicle to the group 'setGroupingState(isGroup)'. If the decision 'getGroupingState = true?' and the vehicle is not listed as a member 'getMemberState = false?', then the last step is to color the vehicle with the group color 'colorVehicle'. The number of group members is counted and the member status is set for each vehicle 'count_group_member_+1'. Lastly, the optimal speed 'AKIVehTrackedModifySpeed' is transmitted to AIMSUN and then Plan 2 stops.

Based on previous experience with AIMSUN, the focus should shift more to decentralized autonomous agents, but still be controllable for traffic management.

6.5.2 Decentralized Hierarchical Groups

In contrast to centralized vehicle groups, the implementation of decentralized autonomous vehicle groups depends on the emerging automotive technologies of intelligent vehicle computer systems which are implemented with the agent paradigm and communication, as well as exchanging information provided by traffic management.

The individual vehicle is benevolent and cooperative, which implies that grouping is seen as something positive and beneficial for the individual and the traffic system. Decentralized vehicle groups are formed on the base of the initial driving preferences of each single vehicle and group plans are accepted and conducted if the group potentially improves the driving, but the individual preferences are still accessible when making new decisions. The autonomous vehicle should be free to join and leave groups depending on its preferences.

The grouping is decentralized concerning the level of vehicles, and hierarchical, because vehicles are grouped with a group leader and members.

ATSim ATSim combines the traffic simulator AIMSUN with the agent interpreter (known for good communication) JADE using a CORBA middleware (the architecture is described in Section 4.3.4) and develops the group-oriented driving method, which respects individual driving, decentralized dynamic vehicle grouping and conflict detection with global coordination. This Subsection is based on a diploma thesis and joint work with `Hung Chu` [64] and mainly on the conference paper of `Görmer and Müller` [140]).

The scenario is kept simple with an artificial setup consisting of three lanes as described in Subsection 6.1 and illustrated in Figure 6.13. There is a group conflict of faster vehicles (z's) driving in lane 3 which are blocked by slower vehicles (x's) on lane 2.

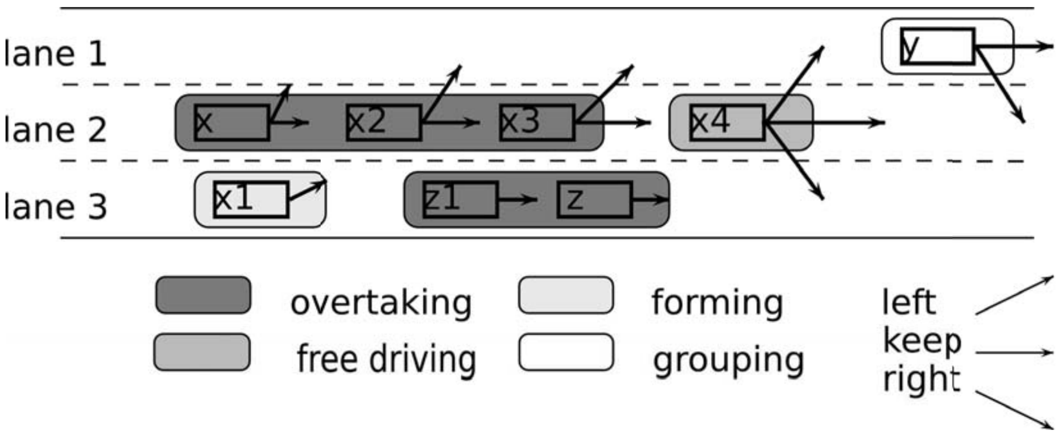


Figure 6.13: Two vehicle groups $G_x = [x, x1, x2, x3, x4]$ and $G_z = [z, z1]$ are in conflict: `Görmer and Müller` [140] p. 4.

Thus, the aim of group-oriented driving (referred to as GoD in [140]) is the decentralized coordination of autonomous vehicles which uses their communication ability so that fast vehicles are not blocked by slow vehicles. Communication between vehicles requires specific message protocols. For all messages exchanged between vehicles, the following simple message format [253] is used:

$$\texttt{msg(id,}id_s\texttt{,}id_r\texttt{,content)}$$

where `id` is the identifier number of the message. id_s identifies the sender and id_r identifies the receiver. The information exchanged between vehicles is the content of the message. Wireless communication mainly allows a vehicle to communicate with others within a limited communication area, which is called Range of Perception (RoP). Vehicles can extend their RoP when in a group by multi-hop communication.

For purely centralized vehicle grouping, extra infrastructure participants in the form of central coordinators need integration into the traffic network. These would communicate with vehicles and classify them into groups. However, due to limited coverage by wireless communication and shadowing effects, a coordinator can not communicate with all the vehicles of a traffic network. Using the centralized approach, it is expensive to apply sub-coordinators to a large traffic network. According to those presented limitations, existing traffic infrastructures should be used, due to the focus on vehicles only for group-oriented driving. Future scenarios should make use of V2X communication.

The focus of ATSim group formation is on a hierarchical decentralized approach. Group formation is obtained by first, decisions and interaction of individual autonomous vehicles and, second, by coordination of vehicles at group level. Group formation is the first step of the group-oriented driving method. A vehicle group G_x with a leader $x(id_x, a_x, d_x, ds_x)$ is defined as a set of vehicles y with the following properties:

$$\forall y \in G_x, f(x, y) \leq \alpha_x$$

where the alpha value α_x is a fixed value predefined by the leader x. The attributes id_x, a_x, d_x, ds_x are identifier number, maximal acceleration a rate, maximal deceleration d rate, and desired speed ds of the leader x, respectively. The function $f(x, y)$ needs to be robust regarding changes in the environment i.e., when the vehicle y is a member of the group G_x in one time step t, it is a member in the next time step $t + 1$. Additionally, the weighting function must be flexible for the needs of the leading autonomous vehicle x, in order to find the desired members at each time step t. The ATSim approach considers the vehicles of the leader $x(a_x, d_x, ds_x)$ and possible member $y(a_y, d_y, ds_y)$ as two points in a coordinate system. A known method is to calculate the dissimilarity between the vehicles of the leader x and others y in terms of their maximal deceleration rate, maximal acceleration rate, and desired speed. The dissimilarity between the leader x and the possible member y is the measured distance between them, and those distances, taken together, are called a metric in the set. Hence, any vehicle y will be accepted as a member by the group leader x if the dissimilarity $f(x, y)$ is smaller than or equals its alpha value α_x.

Depending on the three attributes (acceleration, deceleration and desired speed) as inputs for the dissimilarity function $f(x, y)$, a group leader x can decide based on attributes and the weighing factor (w) whether a vehicle y can join its group. Due to the fact that no centralized device groups vehicles, an autonomous vehicle x must decide and compute the function. There is a maximal acceptable difference $w_{a,x}, w_{d,x}, w_{ds,x}$ assumed for each attribute a_x, d_x, ds_x of the leader x. Therefore, the maximal acceptable differences denote that the leader x is willing to group with vehicles y, whose attributes a_y, d_y, ds_y are in the respective areas of acceleration $[a_x - w_{a,x}, a_x - w_{a,x}]$, deceleration $[d_x - w_{d,x}, d_x - w_{d,x}]$, and desired speed $[ds_x - w_{ds,x}, ds_x - w_{ds,x}]$.

The ATSim idea is to use a normed vector space which is a metric space defined by $d(x, y) = \|y - x\|$. The Manhattan norm results into the Manhattan distance. The distance between any two points, or vectors, corresponds to the coordinates as the sum of the differences. Thus, the Manhattan distance function (MDF) is employed (in Equation 6.1), which calculates dissimilarities between vehicles:

$$f(x, y) = \alpha_1 \frac{|a_x - a_y|}{w_{a,x}} + \alpha_2 \frac{|d_x - d_y|}{w_{d,x}} + \alpha_3 \frac{|ds_x - ds_y|}{w_{ds,x}} \tag{6.1}$$

The alpha values α_1, α_2, α_3 are weighing parameters which identify the importance of the different attributes. For example, the vehicle characteristics of acceleration and deceleration may be, on one hand, less important than the desired speed, which could group more vehicles with that characteristic for heterogeneous vehicles or, on the other hand, more important for grouping vehicles with the same vehicle attributes, so-called homogeneous vehicles. The choice of values of the alpha values α_1, α_2, α_3 must satisfy the condition in Equation 6.2:

$$\alpha_1 + \alpha_2 + \alpha_3 = \alpha_x \tag{6.2}$$

This thesis and the ATSim approach do not focus on optimal values for the alpha values α_1, α_2, α_3, α_x. Values are equally distributed between the attributes and results are given in Chapter 7. Future work should optimize those values, but for the investigation into whether vehicle groups in general improve rush-hour traffic in cities it is out of scope.

Group-oriented driving allows vehicles to form groups based on their similar characteristics, i.e., desired speed, and all members know their group lane. The group leaders communicate about the arranging of group lanes. Figure 6.13 illustrates a conflict showing that fast groups are blocked by other slow groups. By driving in group lanes, potential conflicts between slow and fast vehicle groups can be avoided and for group action also resolved.

The main elements of group-oriented driving are listed and described in more detail in the following paragraphs (compare `Görmer and Müller` [140] p. 2ff.):

1. decentralized dynamic vehicle grouping
2. conflict detection and global coordination
3. collaborative gap solution
4. individual driving strategy of vehicles.

Decentralized dynamic vehicle grouping is based on the parameter of desired speed where vehicles autonomously form groups. A vehicle group consist of a group leader and its members. The group leader is responsible for the coordination of the group's members in order to avoid other slower or faster groups when a *conflict situation* is detected. Since traffic is a dynamic environment, vehicle groups are dynamically created and maintained. This means the number of vehicle groups and the number of members of a group change constantly over time, which is a very decentralized group concept.

The decentralized grouping algorithm is described in Figure 6.14. It starts with an individual vehicle y that wants to group depending on its attributes and that decides whether to join or create a group or, if nothing matches, to drive individually. As a first step, group information is requested by the individual

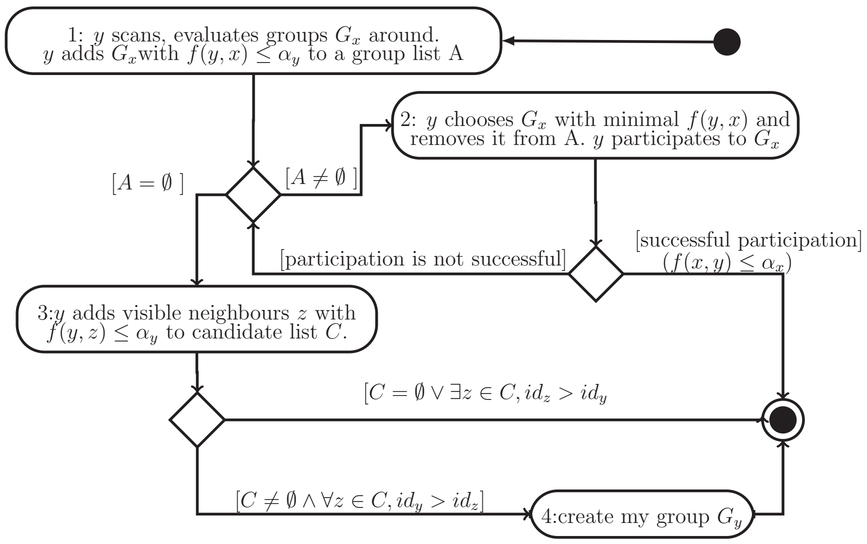

1: y scans, evaluates groups G_x around. y adds $G_x$with $f(y,x) \leq \alpha_y$ to a group list A
2: y chooses G_x with minimal $f(y,x)$ and removes it from A. y participates to G_x
$[A = \emptyset]$
$[A \neq \emptyset]$
[participation is not successful]
[successful participation] $(f(x,y) \leq \alpha_x)$
3:y adds visible neighbours z with $f(y,z) \leq \alpha_y$ to candidate list C.
$[C = \emptyset \vee \exists z \in C, id_z > id_y$
$[C \neq \emptyset \wedge \forall z \in C, id_y > id_z]$
4:create my group G_y

Figure 6.14: Activity diagram of Group Formation.

vehicle y to potential leaders x and every neighbor vehicle. It sends them messages in the following form `msg(id,`id_y`,`id_x`,reqGinfor)`. After receiving all reply messages (`msg(id,`id_x`,`id_y`,`G_x`)`), the potential group member y uses the dissimilarity function $f(y,x)$ to find a suitable group ($f(y,x) \leq \alpha y$). If a vehicle group G_x is found, the vehicle y sends a request (`msg(id,`id_y`,`id_x`,reqPar)` to the leader x of G_x to ask for participation. Group leader x uses $f(x,y)$ to decide if the vehicle y may join its group or not. In case of $f(x,y) \leq \alpha_x$ (or $f(x,y) > \alpha_x$), the leader x replies to the asking vehicle y with a positive (or negative) response message `msg(id,`id_x`,`id_y`,yes|no))`. In case of a negative message for participation, the individual vehicle y will open its own group (step 4) and waits other vehicles joining. However, the creation of a new group requires the fulfillment of the following conditions:

1. y is not member of any group.
2. y knows about at least one candidate vehicle z with $f(y,z) < \alpha_y$.
3. y has the greatest id compared with the id of other candidate vehicles.

The group creator takes the role of group leader. The leader regulates the number of members, but if no participation requests from candidate vehicles are received or all members leave the group, the group dissolves. Maintaining an empty group does not allow the leader to participate in other groups. As a result, after some time the leader should remove its empty group, in ATSim after three simulation steps.

Conflict detection and global coordination is the second element of group oriented driving and coordinates conflicts between vehicle groups.

A situation where a group of fast vehicles are blocked by group of slow ones driving ahead is considered a conflict between vehicle groups. It can be detected by a leader or members. This means, all members x_a of a vehicle group G_x

maintain the information of their group. At each simulation step, each member x_a sends a `msg(id,`id_{x_a}`,`id_z`,reqGinfor)` message to its neighbors z in order to find other groups. This means that each grouped vehicle scans its range of perception (RoP) for other groups which potentially will block its group (conflict group detection). Receiving the neighbors response, x_a uses the following binary decision function 6.3 to determine conflict between his own G_x and the neighbor group G_z.

$$conf(G_x, G_z) = \begin{cases} yes & \text{if } ds_x - ds_z > w_{ds,x} \wedge \\ & \exists z_n \in G_z, p_{z_n} < p_{x_a} \\ no & \text{otherwise} \end{cases} \tag{6.3}$$

where p_{x_a} and p_{z_n} are positions of x_a and z_n on the street. Once a conflict is detected $(conf(G_x, G_z))$, the detecting member x_a sends the information of group G_z to its leader. A group leader can delegate its members to *group lanes*. Group lanes are reserved only for members of a vehicles' group. The choice of group lanes is based on the following two criteria:

1. Minimize lane changes of members.
2. Assure that fast groups are not blocked by slow groups.

The dominance method is used to determine group lanes of conflict groups. Here another negotiation method could be implemented. Suppose that the specific lane on the street h_n where n is the lane of street h and G_x is in the following conflict: $conf(G_x, G_{z_1}), ..., conf(G_x, G_{z_n})$. The following function 6.4 calculates the dominant value $v_{x,n}$ of G_x on lane h_n:

$$v_{x,n} = num_{x,n} - \sum_{i=z_1}^{z_n} num_{i,n} \tag{6.4}$$

where $num_{x,n}$ is the number of members of G_x on lane h_n. The leader of a fast group is allowed to choose its group lane first by definition. Calculating dominant values of its group for all lanes, x chooses a lane with $v_{x,n} \geq 0$ as its group lane. If all dominant values are negative, a lane with a maximal dominant value is chosen. The group lanes of x can not be chosen by other conflict (slower) groups $G_{z_1}, ..., G_{z_n}$, thus, are marked as *busy*. Leader vehicle x publishes its group lane to members and leaders of conflict groups via message `msg(id,`id_x`,[`$id_{z_{1..n}}$`,`id_{x_a}`],infGlanes)`.

Consider the example in Figure 6.13. Assume that a fast vehicle group $G_x = [x, x1, x2, x3, x4]$ and a slow group $G_z = [z, z1]$ are in conflict $conf(G_x, G_z)$. Dominant values of G_x for lanes $1, 2, 3$ are $0, 4, -1$, respectively. Thus, the leader x chooses lanes $1, 2$ as its group lanes. Note that the group lanes assure that members of fast groups are not blocked by members of slow ones.

Communication limitations only allow a vehicle to communicate with other vehicles when they are in a fixed RoP. This means a leader cannot exchange a message with the leader of a conflict group if the two are out of communication range. However, the assumption is that group members can forward the messages of their leaders.

Collaborative gap solution finds a communicative solution for lane changes when gaps are not existing or too small. Generally the driver needs to find a large gap for safe lane changing (gap problem). In order to avoid conflict situations as described previously, group lanes are used to coordinate a vehicle group. Participating vehicles desire to change their current lane to the group lane if they are not already driving in it. Using communication is the key for coordination. An autonomous vehicle can coordinate itself with others to create its own gap for lane changing. The gap problem is illustrated in Figure 5.6.

Assume that vehicle V_c wants to change to lane 1. Each vehicle V_a, V_b, V_c needs a distance to stop, which is illustrated by the length of arrow in front of them (see Figure 5.6). Let $s(V_a, V_c)$ denote the gap between V_a and V_c. $s(V_c, V_b)$ is the gap between V_c and V_b. In normal traffic, V_a always drives at a secure speed (whereas in a group it would drive with a minimal gap since a warning could be communicated), which allows him to stop behind V_b. Using the car-following model of `Gipps` [132] this speed can be calculated. As shown in Figure 5.6, the stop position p_{V_c} of V_c is behind the stop position p_{V_a}. This indicates that, if V_c changes to lane 1 and decelerates its speed, it would collide with V_a. The following conditions must hold for V_c for safe lane changing:

1. The rear gap $s(V_a, V_c)$ should be big enough to allow V_a to stop behind V_c.
2. The front gap $s(V_c, V_b)$ should be big enough to allow V_c to stop behind V_b.

Driving strategy of individual vehicles: the vehicle chooses lanes for reaching its goal as fast as possible for the next time period using the utility function. Note that reaching and maintaining desired speed is the original goal of each vehicle. However, a group member should obey the coordination of its leader to avoid conflict situations. Thus, a vehicle should always decide whether to choose its lane based on the coordination of the leader or based on its local goal. Group-oriented driving provides a lane choice strategy based on the state of a vehicle using four states:

1. *grouping:* The vehicle is driving individually (not grouped) on a lane, trying to fulfill its desired speed with a utility-based decision.
2. *forming:* The vehicle belongs to a group, but does not drive on the group lane. The vehicles ignores its own utility in favor of the coordination of the group leader and follows him. If necessary, the vehicle accepts to reduce its current speed in order to change to a group lane.
3. *overtaking:* The vehicle belongs to one (stable) group and is driving in a group lane which can change. If the utility is better in another lane, it will change to the group lane, which maximizes its utility. Utility-based decision making is used by the vehicle only in its group lanes.
4. *free driving:* The vehicle is a group member and drives in front of all others including the leader. Thus, it can ignore the coordination and uses utility-based decision making for choosing its future lane.

The utilities-based function capacitates a vehicle x to accelerate to its maximum speed and avoids blocking its following vehicles while choosing the preferred lane n. The following utility values of vehicle x in a lane n were used:

$$U_{x,n} = [V_{s,n}^{new} - V_{s,n}^{current}] + [V^n ew_{x,n} - V_{x,n}^{current}] \tag{6.5}$$

$V_{s,n}^{new}$ describes the 'potential speed' of the follower of vehicle x in lane n. $V_{s,n}^{current}$ is the 'actual speed' of the following vehicle of vehicle x. If vehicle x changes to lane n, then $V^n ew_{x,n}$ is its potential speed. The current speed of x in its current lane is $V_x^{current}$. The `Gipps` [132] car-following model was employed for calculating the speed V of vehicles.

Detailed listings are in the Appendix E.1 representing the JAVA class for the vehicle agent written with the FIPA communication standard, which basically functions step-based and implements the `Gipps` car-following model and calculates its speed.

The ATSim approach is very detailed in respecting different aspects of coordination: individual driving, decentralized vehicle grouping and conflict detection with global coordination. In future those aspects should find application in mixed dynamic traffic groups implemented with MATI.

6.5.3 Mixed Dynamic Groups

The idea of completely decentralized and dynamic vehicle groups is that individual vehicles can gather all the necessary information for coordination purposes through V2V (vehicle to vehicle) or V2I (vehicle to infrastructure) communication. Here, more research is required, for example Hovering Data Clouds (HDC) by `Axel Wegener` [164, 371, 372], which could be static HDC on a traffic light and a dynamic HDC as an information base for vehicle groups. On a good information base, autonomous vehicles can decide what actions to take or provide the driver with relevant information and choices which are advantageous for the vehicle's preferences. That implies an automated (or half-automated) intelligent board computer including communication and information gathering, which can calculate the best options and can coordinate the vehicle with respect to others and the infrastructure preferences for optimization purposes. In this thesis, completely autonomous vehicles coordinate their behavior in a simulation. Different penetration test have been performed in relation to the communication frequency. For future work different degrees of penetration of equipped automated and half-automated vehicles can be conducted to investigate the influencing effects.

Communication is necessary in order to form groups. A vehicle must retrieve the group information from the leader or an information access point provided by the TMC, calculate the similarity and then send a request to join to the group with the most similar parameters. There are two ways in which the agents (which can represent vehicles, traffic lights, streets) can communicate: centralized or decentralized.

In the centralized approach, the agents do not talk to each other directly. Instead, they are all connected to a central node which controls the agents. In

Streetworld 4.3.1, the first implementation of the group forming process is based upon the centralized approach. The centralized approach has a coordinator with which every agent is connected. The coordinator is an agent itself. Its id is propagated to the agents during the initialization.

In the decentralized approach, every vehicle is sender and receiver at the same time (V2V communication). Vehicles can establish connections with each other on their own.

All scenarios mentioned in Subsection 6.1 can be accessed and simulated in MATI with the SUMO environment and Jason agents and HDF5 tool.

Two groups are formed in MATI: static compulsory vehicle groups and decentralized dynamic vehicle groups. A group is a collection of vehicles that share the same or similar parameters.

Static Compulsory Groups

Centralized grouping is divided into three main phases as mentioned before in Subsection 6.5.1:

- Pre-phase: discovering groups, calculating similarity, joining groups, creating groups
- Main phase: performing group activity
- Post-phase: adjourning

In the *prephase*, a group is formed based on parameters which define a vehicle in the SUMO environment. Some of the parameters can change during run time and some cannot.

The contract net protocol (also used in variation for AIMSUN) is adapted for Jason agents and listed in the following.

Overview

```
Initialization
/** init_vehicle.asl */

receiver(coordinator). // initial belief
!init.
+!init

<-
.my_name(MyName);
.concat("vehicleArtifact_",MyName,VArtifactName);

lookupArtifact(VArtifactName,VArtifactID);
focus(VArtifactID);

!!start;
.
```

Listing 6.7: Centralized Group Formation with Coordinator.

The belief receiver (coordinator) can then be used to send messages. For example:

```
?receiver(C);
.send(C, askAll, groups(X), List1);
```

Listing 6.8: Coordinator sending messages.

Here the wording is agent-centered: Every agent has its own vehicle artifact through which the agent is connected to the SUMO vehicle. Perceptions from the SUMO environment are mapped to the agent beliefs. There is a detailed listing in the Appendix C.2.1.

The perceptions from the SUMO environment are marked as observable properties which means they are integrated in the agents' belief base. For example, an agent can access the speed parameter at any time:

```
%?speed(CS);
%.println("Current speed is", CS);
```

Listing 6.9: Current Speed.

The coordinator will create a coordinatorArtifact that the agents will focus on. It is used to find and store groups.

```
public class CoordinatorArtifact extends Artifact{
private List<String> groups;

@OPERATION public void init() {
groups = new LinkedList<String>();
defineObsProperty("groups", "no_groups");
}
@OPERATION public void addGroup(String grpName) {...}
@OPERATION public void findGroups(OpFeedbackParam<List<String>> opGroups) { ... }
}
```

Listing 6.10: Coordinator Artifact.

Decentralized Dynamic Groups

In MATI there is no upper limit for parameters, so the parameter can also be greater than 1. This is no problem, because if the return value gets too big, the vehicle simply will not join the group. The alpha parameter is used to weigh the parameters. The desired speed is more important for forming a group than the acceleration of the vehicle.

The following parameters are selected in the context of this thesis, but can be varied depending on the purpose of investigation:

- desired speed: the vehicle aims for driving at the favored speed, i.e., the mandated speed and additionally 10% faster.
- acceleration: the speeding-up of a vehicle depending on its characteristics.
- deceleration: the breaking and stopping performance of a vehicle measured in m/s.
- route: the connected streets the vehicle plans to drive on in a street network.
- route costs: streets have weights based on experience of historic data, i.e., travel and stop times, charges, fuel or energy consumption, CO_2, etc.
- position: the actual place the vehicle is located.
- destination: the desired place the vehicle wants to reach.

Position, destination, route costs and the vehicle characteristics acceleration and deceleration are static parameters which do not change during simulation. The other parameters (desired speed, route) are dynamic, which means they do change during simulation. Agents perceive changes in these parameters and can react to them. The current edge is used to get near to groups which are driving on the same street (edge) as the vehicle which is trying to join a group. Group members should have a similar target and speed. The similarity is calculated with the Manhattan Distance Function (or Euclidean Distance Function).

Using the Manhattan Distance Function (MDF), an agent who is not in a group compares his own parameters with the ones from the group and chooses the group with the closest similarity. The MDF calculates the similarity between a vehicle and a group. It compares each parameter and adds up the difference between the values. The factors in the nominator are used to normalize the parameter to a value between 0 and 1.

Decentralized group formation depending on the preferences was done first with the help of the Manhattan distance function (MDF - also used in a simpler version in ATSim) and later was improved with an adapted Euclidian distance function (AEDF - used for MATI), described in the following paragraphs.

6.5.4 Individual versus Group

Individual driving is basically a tactical decision acting on the environment directly like driving straight, turning and accelerating or decelerating.

Group behavior maximizes the individual performance with strategic decisions, but has a coordination and communication overhead formulated in Equation 5.39. The overall traffic flow is enhanced with group formation due to a set group speed and less safety distance than driving individually, because the autonomous vehicles can drive in a coordinated and organized fashion (in contrast to human drivers, which have physical and psychological influences i.e., reaction time and attention loss).

Today, individual driving is very prominent, but this could change due to emerging traffic technologies including autonomous driving with vehicle groups. Group formation and their coordination is an interesting process for traffic management because it can double or triple the throughput as stated in `Lioris et al.` [231]. Also for individuals, vehicle groups are beneficial because of reduced travel times (how to exploit travel times in urban traffic is described in `Groß et al.` [143]) and other costs. This may be investigated with different penetration levels of equipped groupable vehicles in future research building on this thesis.

Aspects of the quality of travel and stop times from the local and global perspective are investigated in the ATSim approach. The MATI concept tries mixed groups whereas decentralized vehicle groups are more beneficial than static centralized groups. AIMSUN approaches the global perspective with fixed static groups.

Group formation aims to improve the travel and stop times in rush hour traffic. Individual driving seems to be advantageous for light traffic density where the throughput measured in travel and stop times is in a good service quality of LOS A and B, i.e., nighttime travels. Individual driving does not have extra coordination expenses, but this does not count too much, because grouping is done at red traffic lights where all vehicles need to stop or, in MATI, decentralized, which is more a matter of calculation of individual autonomous vehicles. With increased traffic density of LOS C to E, autonomous vehicles groups can improve the throughput and thus reduce travel and stop times due to coordinated speed and gap organization within vehicle groups. Therefore vehicle groups are more advantageous in rush-hour traffic, whereas, in traffic jams, new coordination concepts with groups and communication could avoid or dissolve them faster.

Autonomous groups reach their goal more efficiently due to a common speed and smaller gaps than individual vehicles.

Individual vehicles in ATSim can always decide whether to choose their lane based on the coordination of a leader or based on their local goal. Individually, the vehicle tries to fulfill its desired speed, making a utility-based decision. In a group, the individual vehicle can drive in front of all the others, ignoring the coordination and using utility-based decisions for choosing its future lane instead.

6.5.5 Cooperative Groups

Intergroups (groups compared to groups) are not considered in the centralized approach: groups just form for each red phase and depending on the direction they are going. There is no solution if conflicts occur.

ATSim groups use the group-oriented driving method which respects individual driving, decentralized vehicle grouping and conflict detection with global coordination. Especially the last aspect is important for groups in order to resolve a group conflict and coordinate themselves using negotiation or the dominance method used here.

MATI's decentralized vehicle groups are automatically cooperative groups: they merge if the values are within their weighing factors, groups dissolve if they are outside their values of desired speed, acceleration etc.. Thus, autonomous vehicles can automatically join and leave groups. Conflicts are not considered. Future research should implement group conflict aspects and solutions in MATI.

6.6 Summary

The claim in the Simulation Section 6.1 that MATI simplifies the integration of JAVA-based MAS which support BDI architectures with discrete environments of (traffic) simulators is demonstrated in this Chapter 6. AplTk was extended and EIS used, which makes the connection of an environment with one or several agent interpreters very comfortable and uses mature simulators for small-scale but detailed systems. The environment can be extended with organizational aspects such as cooperative dynamic vehicle group coordination whereas the interpreter should also embody interaction features for autonomous coordinated actions.

Present traffic management is extended by group formation for speed and gap adaptations, which routes targeted and selective traffic flows individually or in convoys of vehicles.

The decentralized approach has its challenges because of communication, calculation and coordination overhead. In the preliminary phase, the determining factors are the individual reaction time, the vehicle properties, communication and coordination especially when automatizing decentralized vehicle groups. A known measure, the time span of braking distance for each vehicle, was used as an equivalent for identifying similarities to other neighboring vehicles.

Agent implementations are presented and evaluated. Interaction through communication in AIMSUN and ATSim versus reacting to the environment and to the behavior of others is outlined with MATI, programmed in JAVA or AgentSpeakLanguage internally and reacting to environment artifacts.

The communication-based group formation in ATSim is a cooperation of vehicles for forming groups, solving *conflict situations* and coordinating individual driving and gap problems.

Triggers of group formation are usually globally motivated in form of high traffic density, LOS C to E, or selected sections or traffic lights for grouping by the TMC. Group formation can be influenced by the length of the street sections or red phases of the traffic lights due to their coordination overhead. Intergroups and individual vehicles with different penetration rates of group equipment could be addressed more in the future.

The group formation methods aim to show that it is better to drive in a convoy of vehicles - a vehicle group with the same speed and smaller gaps - than individually, just like with AHS for more throughput. Individually, it is better to drive at night time or in light traffic density, LOS A and B. The group formation overhead was reduced by strategic decisions of grouping at red traffic lights or just using calculation and as little communication as possible. For traffic jams, new coordination strategies are needed with groups and communication, which could avoid or dissolve them faster.

The different group characteristics of centralized and decentralized groups were considered. Group size is also dependent on communication, red phase and the coordination overhead; different groups of homogeneous and heterogeneous vehicles are simulated in different urban networks.

Everything that can be counted does not necessarily count; everything that counts cannot necessarily be counted.
Albert Einstein (1879-1955)

Chapter 7

Evaluation and Presentation of Results

In this chapter, the models which are described in Chapter 5 are evaluated experimentally. For this we use the MATI simulation platform presented in Chapter 4, as well as the implementations of the scenarios in Chapter 6. The last phase verifies the model for autonomous group formation with the assessment of the simulation data. The effects of the grouping models which use selective routing for physically coordinated driving are investigated with regard to whether it is better for the individual and global network to drive individually or in a convoy of vehicles with or without group formation on the same route. What are the influencing factors for manipulating the convoy of vehicles and their coordination?

The central question is: Can autonomous vehicle groups contribute to traffic efficiency, i.e., throughput?

The following focuses are important:

- the global view from the perspective of traffic management,
- the group view from the perspective of joint plans and goals of coordinated members, and
- the individual view of each single traffic participant.

The group efficiency depends on the following influencing factors and variable parameters:

1. Realistic scenario locations: the Hildesheimer (HI) scenario, which is a proper subset of Hanover Südstadt (HS) with a decentralized dynamic algorithm based on similarities for grouping vehicles.
2. variations of vehicle properties: homogeneous simulations versus four heterogeneous types in which the standard type from homogeneous simula-

tions plus three others (small, fast and long) are also integrated using probabilities common in urban traffic.

7.1 Research Questions and Hypotheses

The purpose of this thesis is to explore the association between dense urban traffic and autonomous vehicle groups. The area of concern are vehicles moving slowly through urban cities and causing negative effects for drivers and cities. Conditions that could be improved are, amongst others, the better use of travel time through the use of autonomous vehicles and lowering emissions through AVs which use additional coordinated driving methods including reduced travel and stop times and an improved use of available street capacities with smaller gaps between vehicles. Difficulties that need to be surmounted are the shift from individual to group driving, vehicular and infrastructure information exchange and the enhancement of existing city traffic. Questions seeking answers result in hypotheses which can be measured with different metrics. Subsequently, the research question will lead to its hypotheses, then the metrics can be validated.

The following subsections encompass the relevant research questions with the expected hypotheses as outcomes, which need verification with the metrics in the experiments performed.

7.1.1 Group Effectiveness

Group effectiveness is especially measured in the environment. On highways, in the context of convoys and platoons, some of the benefits were reduced congestion, substantially shorter commutes during peak periods and greater fuel economy. Those effects would be an enrichment for urban traffic and were tested in traffic simulation systems with autonomous vehicles. Instead of individual traffic there is cooperation and coordination for improved use of available infrastructure resources such as street lanes.

Research Question

The first essential research question envisions the effects of groups in urban traffic: How would future urban traffic look like with AVGF and can they contribute to traffic efficiency?

In order to answer this first research question, some assumptions need to be made. New technologies like vehicular communication and autonomous vehicles are assumed with a 100% penetration rate. Therefore, there are no driver-driven vehicles. Since this ideal does not exist in realistic urban traffic nowadays, traffic simulation systems need to handle realistic data to simulate future traffic. Traffic simulation systems are either macroscopic or microscopic as described in the Background Chapter 2, but for this thesis only microscopic traffic simulators and their driving and lane changing models are considered. Additionally, the individual behavior of vehicles is enhanced with multi-agent paradigms for representing autonomous vehicles in their decision making, which is essential for grouping. Both simulation systems need to include models which are as realistic as possible to be able to transfer the results to future traffic on the streets. Therefore, coordinated driving is implemented based on similarities of vehicles and their route preferences with less gaps between vehicles. The intended positive effects of the autonomous vehicle groups are more flow, green waves, and less emissions.

Metrics

The flow can be measured by throughput and delay for the overall system. In other words, for the individual, the goal is to minimize the travel time from the current position to the destination, but also to minimize the stops at traffic lights in the urban network.

Green waves or optimized networks are provided by the infrastructure and expert traffic engineers which model good signal plans to fit the urban network's needs. This can be implemented in a static, traffic-independent fashion or a dynamic, traffic-dependent fashion. For the simulation, static traffic-independent traffic signal plans were used to inform the autonomous vehicles about the rest time value of each signal phase. The optimized green wave location Hildesheimer Street is used as a green wave scenario, which is compared to the total network of Hanover Südstadt.

Emissions are a hot topic in urban areas with discussion of particulate matter, for example, in Stuttgart, Germany. But fuel consumption and exhaust gases have been discussed lately, for example in the wake of the Volkswagen group scandal. Emissions are difficult to measure, especially for future autonomous vehicles which are not on the market yet. Although simulation programs include different emissions parameters depending on the vehicle type and its travel time, this remains an estimate due to the complexity of the topic and the lack of available data.

Hypotheses

Benchmarks are the throughput and delays for traffic networks, therefore some hypotheses are formulated to address them:

1. If autonomous vehicle group formation is applicable in urban networks, then the throughput is increased and delays are reduced compared to individual driving/driving with no vehicle groups.
2. The more groups are formed in urban traffic, there exists an improved trade-off for travel and stop times.
3. If the vehicle groups are fully equipped with communication and autonomous features, then the performance of AVGF is near the optimal use of the available street capacities in dense traffic.

The optimized green wave location Hildesheimer Street is a proper subset of the total network of Hanover Südstadt and those locations are compared with each other. The environment can be seen as the global infrastructure, but with individual characteristics. Therefore, the individual vehicle types are simulated using a homogeneous standard type and then with four different characteristics: fast, normal, small and long. Then the following hypotheses can be proposed:

4. The standard vehicles perform better in optimized green wave locations than a mix of different vehicle types.
5. Vehicles with the same characteristics, such as speed, stay in vehicle groups longer in group optimized locations (like green wave) whereas diverse vehicle types perform better in larger networks.

7.1.2 Group Similarities

Group similarities represent the organization of the groups and their architecture. Classic traffic management uses centralized architecture whereas this thesis focuses on the decentralized aspects. Thus, a dynamic group algorithm based on similarities of vehicles was developed and used for dynamic group formation. The influence of AVGF within the groups and, in parallel, the inter-group behavior, need to be investigated.

Research Question

The second research question deals with the similarities of the groups: What is the group strategy, the similarities and limitations of AVGF?

In order to investigate the second research question for group similarities, some specifications are determined. There are different architectures which fulfill the predefined requirements. Usually, a centralized architecture is used to support the traffic engineers in regulating traffic with a top-down approach.

This thesis does a headstand and prefers to use decentralized architecture for dynamic vehicle groups. The idea was to compare a static and dynamic group formation algorithm. The static group formation was too simple and not very realistic with the color sort algorithm, contract net protocol and dissolving phase. There are A* or more realistic static algorithms which would fulfill the needs of static but decentralized architectures. Since the static group formation would be condemned to losing from the start, it doesn't make sense to compare the two. Thus, the limitation is to use only the self-made decentralized algorithm on similarities of vehicles. Hence, the preferred group strategy is to use only the dynamic algorithm with the similarity functions and compare the mean distances and variation using realistic locations and individual scenarios with homogeneous and heterogeneous vehicle groups. The differences between groups can be investigated.

Metrics

The mean distance is a numerical description of how far apart the autonomous vehicles (AVs) are over the average of all vehicles. The mean variation is the change in distance between AVs per time step. Inter-group behavior is the key to finding out what the glue between group members is and how it differs compared to the other groups.

Hypotheses

1. If the traffic scenarios are more complex and dynamic, the more dynamic group strategies improve the individual and global performance.
2. Homogeneous vehicles will stay in groups longer whereas fast vehicles can compensate flows in large traffic networks.
3. Homogeneous vehicle groups are more effective in throughput than heterogeneous groups in optimized green wave locations.
4. Group members need to be as similar as possible to each other and as different as possible from other groups.

7.1.3 Group Formation

In this section, the statistic evaluation of the distance data is conducted. For the elementary analysis, a table object (see Table 7.3) was constructed with the previously generated array (see Table 7.2), which simplifies further analysis (see Appendix C.4). In urban traffic, group formation by vehicles is not standard. Usually it is assumed that vehicles drive individually using their own best strategy, where each maximizes its own utility, while obeying the rules and regulations provided and enforced by traffic management. The intention is that the resources available of streets and their lanes be better used through group formation. Group formation has an organizational structure and is not possible without interaction. For this work, group formation is based on similarities of vehicles, comparing 14 parameters.

Research Question

The third research question deals with the shift from individual driving to using the benefits of group-oriented driving: Why are vehicle groups not considered as a model for using the infrastructure and resources of urban networks nowadays? What is dynamic group formation and what are group formation models? What new technologies and metrics can be used for analyzing the effects of AVGF algorithms?

The first two questions lead to the background of group formation while the essential question is the last, answered with metrics and a hypothesis. Since some time, platoons of vehicles have been considered on highways. Also, traffic signals try to manage vehicles into swarms, so that they fit better into the signal phases of green driving and red stopping. Today there is a lack of standard information about all relevant traffic conditions to each driving vehicle. That is why group driving has only been possible in pre-defined platoons like military convoys, where there is a leading vehicle and one which marks the end.

The intention of using dynamic group formation is a bottom-up approach from an individual and decentralized perspective interlinked with communication. Dynamic group formation is designed to be feasible in urban traffic. Therefore, the notion of groups is kept loose without too many boundaries like strong commitment and regulations such as entering and leaving groups after a certain period or landmarks. Each vehicle should be able to enter or leave the group, still being able to maximize their own utility function, but using the group coordination if that promises to be more beneficial. For this, hierarchical role-based group models are considered less than the goal-based group models where each member is considered equal.

Analogous to biology, where swarms have their own organization and intelligence, the concept of decentralized bonding and interaction seems especially feasible with future traffic technologies like vehicular communication and intelligent board computers making decisions for autonomous driving. Vehicular and infrastructure communication provides new possibilities for group formation to exchange actual position, destination goals and route choices and their coordinated actions, also depending on infrastructural regulations. Therefore, communication exchange or interaction is a prerequisite for group formation. Group formation depends highly on the network and the number of total vehicles, which has an effect on the possible number of groups and their group size.

Metrics

The network is relevant as a metric regarding whether there are enough alternatives once a node like an intersection is jammed and rerouting might help to distribute the traffic in the network. Also, the network infrastructure as a source of information and regulation with traffic signals influences group formation. The distances between the intersections are helpful for group formation and coordination, providing free driving without other outside influences. This is why platooning has mostly been done on longer stretches like highways where the outside influence of infrastructure and network conditions does not change.

In the urban network, the density of vehicles is relevant because it indicates the degree of capacity utilization. This is also an indication of how the overload on certain streets could be alleviated by choosing other routes beforehand. This coordination can be done by vehicle groups using the relevant information for avoiding traffic jams.

Hypotheses

1. Are groups are especially useful for crowded, but not entirely congested networks? Are groups less beneficial in low density traffic. In stopped traffic, vehicles can only act as an information exchange, which could also be helpful. With too few vehicles in networks like during nighttime traffic, it might be hard to find a group of vehicles to drive together, thus, it might just have a coordination overhead in terms of time used. This elaboration

also points towards the number of groups being influenced by the total amount of vehicles.

2. At the same time, is it possible that the AVGF adapts to individually desired behavior; in other words, is it dynamic enough to maximize the use of the utilities? Group size depends on the definition, on the algorithm, but also on the network and number of vehicles, which is described above. The group size is highly dependent on the routes between intersections. One assumption is that the urban group size is smaller than automated highway systems consisting of 8 to 25 vehicles.
3. The number of vehicles is one indicator for possible coordination. Different cooperation levels for group formation technologies range from complete communication to only a few well-informed agents and can be measured with different penetration rates, as well as autonomous equipped versus automatic or driver vehicles. Additionally, different traffic demands indicate different amounts of vehicles in the network, such as little traffic versus rush hour / traffic jams, can be compared. Thus, the hypothesis is: The maximum cooperation level and slightly congested traffic in an urban network are the best conditions for using vehicle groups and full available communication is mandatory for group formation.
4. The observed number of groups depends on the network and the number of vehicles available for grouping, as well as the algorithm. Due to the AVGF algorithm, most of the vehicles will be in groups.
5. Group size: A network with lots of intersections including traffic lights and comparatively small street segments between them can reduce the maximum group size, whereas large street segments between intersections will enlarge the possible group size. The observed group size will be between 3 and 15 vehicles.

7.1.4 Group Parameter

Group parameters represent the characteristics with their dependencies in one object of the class or differences between the objects of the same class which are formed as a data cluster to identify a group. Using the example of the card game 'vehicle quartet', the vehicle class has different attributes, like speed (km/h), performance output (HP), capacity (ccm), cylinders, consumption (l), price (EUR) and acceleration from 0 to 100 (s). From the point of view of data analysis, the vehicle quartet is a quantity of objects of the vehicle class. The vehicle group parameters were chosen on the base of available data in a simulation environment for the vehicle class listed in the test environment 7.3.1: max speed, acceleration, deceleration, actual position, desired destination, individual route. Those are compared with Euclid and Frobenius metrics including an alpha value which includes weights (as a standard with the same distribution) and each individual parameter including x and y coordinates for position and destination. Those parameters are clustered into a similarity value of the vehicle properties to match the individual vehicles to similar groups of vehicles.

The complexity rises for other than linear combinations. There are necessary parameters for group formation (same destination or route) and other weaker parameters. If the rout does not coincide, there will be no platoon causing a KO criteria.

Research Question

The fourth research question regards the vehicle parameters and their influence on being grouped: Which parameter variables of individual driving influence the joining of a group?

The criteria for the group parameters were chosen as generally as possible, so they can be fulfilled by each vehicle type. Therefore ,only position data and speed data were used for finding similarities. This data comes from the traffic simulation. Especially in urban traffic, the speed data of different vehicles is quite similar due to the regulations of maximum speed and the fact that each individual vehicle tries to reach the maximum speed during its free driving. The position data of actual position and destination is straightforward in a green wave Hildesheimer location with just a time gap; in the Hanover Süd scenario there are basically two to three starting points at which vehicles arrive and two main destinations for leaving the network. Since the alpha values all have the same weight, no difference is caused by the design decision, but it is a possibility of influencing the groups by each individual parameter.

Metrics

The grouped metrics based on four dimensions for traffic result in one alpha value, as expected due to the Euclid elements. In order to see the individual differences in the parameters, the data is logged for route matching like x and y position, x and y destination, maximum speed, acceleration, deceleration, current speed, and additionally for total and average stops. It is a bit difficult to only take the x and y coordinates of position and destination, since it has a time variable and depth. Depth is not relevant since the autonomous vehicles do not fly (yet). The time can be managed for a point for each time step, thus reducing it to two relevant dimensions. Therefore, the map with the coordinate values needs to be slightly fitted and turned in order to have a reference.

The maximum speed is limited by the infrastructure or the vehicles' individual desired speed. The acceleration and deceleration are different only for varying vehicle types, otherwise constant. The current speed is very variable depending on where the vehicle's actual location is. For instance, the vehicle can be stopped at a traffic light or it can be speeding to catch the last sequence of the green signal phase. The total and average stop times are a parameter for efficiency, since less stops are better for individual and global urban traffic. The effects of the grouping models using selective routing for physically coordinated driving are investigated regarding whether it is better for the vehicle to drive individually or in a convoy of vehicles with or without group formation on the same route.

Hypotheses

1. The influencing factors for manipulating the group of vehicles and their coordination are the actual position and current speed, although by intuition the destination should have the highest impact of all the parameters.
2. How many factors influence the group formation? Which are independent? The alpha values are independent, whereas the average and total number of stops, the current speed, and the actual position are highly dependent on group effects. Other dependent factors, surprisingly to a minor degree, are the maximum speed and destination, although this seems counterintuitive. Acceleration and deceleration and the destination influence the group choice to a minor degree.
3. If there are more variable parameters which can effect the traffic, then the results can be interpreted in more than one way.
4. The driving algorithm with smaller variations in parameter is beneficial for the vehicles, no matter if grouped or individual driving.

7.2 Evaluation Methodology

This experimental evaluation replicates the scenarios depicted in Chapter 6. A heuristic approach with simulation is used to investigate autonomous vehicle group behavior and the associated effects on urban traffic, due to its complexity.

The assessment phase focuses - besides the technical challenges presented below - on the following non-technical issues to vehicle grouping:

- Analyze the impact on infrastructure and environment: compare test results, vehicle group requirements and simulation results on potential impact.
- Assess economic viability and potential policy impacts.

The key functional element is the coordination of vehicles into groups. Technical challenges center around the following items:

- Prerequisites are technical requirements like the use of existing reliable communication and autonomous features of vehicle behavior.
- Develop vehicle group strategies that will allow vehicles to operate in urban traffic networks without changes to infrastructure or environment and enhance traffic flow, i.e., a leader - member role-based algorithm.
- Integration of simulation systems to define scenarios that observe individual behaviors and run smoothly with fixed infrastructure settings like signal plans.
- The simulation and its scenario networks ensures that the vehicle groups perform as designed.

7.2.1 Experimental Design

A methodology for designing experiments was proposed by Fisher in 'The Arrangement of Field Experiments' (1926) and 'The Design of Experiments' (1935), referred to as Fisher's principles, which are followed in general in this thesis.

The design of experiments aims to describe the variation of information under conditions that are hypothesized to reflect the variation. The simulations are true experiments in which the design introduces conditions that directly affect the variation; natural conditions that influence the variation are selected for observation. The experiments aim to predict the outcome by introducing a change of preconditions, which is reflected in a variable called the predictor (independent). The change in the predictor results in a change in the second variable, hence called the outcome (dependent) variable.

The design includes the selection of suitable predictors and outcomes. A dynamic group formation algorithm based on similarities of vehicles was developed. The single parameter for group formation is the predictor and the outcome is the formation of autonomous vehicle groups in simulation.

Another important design aspect is the delivery planning of the experiment under statistically optimal conditions given the constraints of available resources. This is taken into account by repeating simulation runs of the same setting with random start configurations. The configuration for all realistic scenarios (HI and HS) is to use homogeneous as well as heterogeneous vehicle groups and assess the impact on group formation. Several scenario runs are performed for representative evidence with randomly generated starting configurations (about 10 runs). Constraints are the time available for simulation runs and computer resources for executing the simulation.

The establishment of validity, reliability, and replicability is a necessity in research. The results are described in Section 7.4. The interpretation of those regarding aspects of interest or for different stakeholders is discussed in Section 7.5. The code and analysis procedure are made available in the Appendix C.4. This thesis and published papers provide a sufficiently detailed discussion and documentation with statements about: (a) group formation contributing to traffic efficiency with faster travel time and less stop times, (b) penetration rate with communication and (c) interdependencies of AVGF regarding global, group, and individual aspects.

7.2.2 Evaluation Criteria

This thesis is analyzed on the basis of the research questions and their metrics, described in Section 7.1 and, in general, with five common key criteria:

1. The criterion of *relevance* is used to evaluate the priority of this thesis: an important function from a developmental perspective, according to Chapter 5, with respect to the model of the group life cycle. Additionally, the validity of the results is answered by the case study in Chapter 6 and the

evaluation, described here, answers the question if the design fundamentally suits the goals associated with the thesis to make a contribution with grouping autonomous vehicles in urban traffic.

2. The criterion of *effectiveness* is used to assess whether the approaches proposed in this thesis achieved its goals. Thus, the targeted and actual outcomes are compared, and the thesis' goals should be expressed including any unintended positive or negative secondary effects or consequences that can be observed in the evaluation. This includes the quality of the introductions and implementations, regarding whether the surrounding conditions are given in order to successfully realize the autonomous vehicle groups and whether they are implemented as described in the model. The internal group quality can be measured with a similarity parameter and the distance function including alpha-values. The inter-group proximity is measured by the number of groups.

3. For the criterion of *efficiency*, a cost-benefit analysis is used to assess the thesis. The economical use of resources is the central issue expressed with the cost-benefit-ratio. Efficiency includes other available methods which could have achieved similar results. Here, the autonomous vehicle group performance for the global, group, and individual use is assessed regarding whether autonomous vehicle groups are accepted (satisfaction). The measures for global use are the simulation duration and the number of vehicles. Also, the maximum or optimum can be respected and where the simulation crashes. For the group use the measures are the destination, the distribution of vehicles in the network, the travel and stop times for the route from A to B, the throughput of vehicles in a signal phase, and the traffic density. For the individual use, the travel and stop times are an indicator as well as egocentric default versus cooperative group behavior. Thus, the technical proximity, e.g., performance/efficiency of km or duration, stability (without jitter versus dissolving of groups), coherence (how long is a group together, penetration rate), communication efficiency (number of messages sent, penetration rate), and runtime, is analyzed.

4. In addition to the thesis' direct goals, there is also the *overarching impact*, the big objectives that are the reason why the decision was made to promote the thesis in the first place for example the impact on urban traffic through improving the flow by grouping vehicles. The question is whether the concept and evaluation of the vehicle group effects is constructed to be sustainable. Often it is not possible to measure overarching impacts. In such cases a plausibility check and an estimation of circumstantial evidence provides an indication.

   The aim of this thesis is to show future effects of urban traffic, providing valuable input for the future of mobility. *Sustainability* combines efficiency and safety with environmental friendliness and social compliance. The first is met, but the latter will need further details of investigation.

7.2.3 Tools and Measuring Equipment

The tools are included in the traffic simulator, but also extended for some specific measurements. Measurements need to be executed and logged, so that an analysis can be performed:

- measured data are presented individually in paramstables, differences in distables and groups in grouping tables
- GroupingBenchmarkTool includes a benchmark table (CBenchmark apltk-common), including total group size and maximal group members
- travel and stop time for default and dynamic groups
- simulation time is fixed for one hour for comparisons using fixed signal plans in rush-hour traffic (LoS C or D)

Therefore:

1. An argumentation of decentralized grouping with dynamic group algorithms in different dimensions is undertaken.
2. The claim is that the hypotheses are valid, The main research question, namely whether vehicle groups improve efficiency, measured by travel time, should be answered.
3. Environment use cases are executed with two different coordination methods and agent variations.

7.2.4 Dimensions

After determining that the group-oriented driving could manage as much traffic as the default simulation, the experiments were extended along two main dimensions. In particular, the trade-off in terms of computational complexity versus performance level was characterized.

First, the investigations studied two realistic scenario locations, the Hildesheimer (HI) scenario being a proper subset of Hanover Südstadt (HS), with a decentralized dynamic algorithm based on similarities for grouping vehicles.

Second, the impact of the main parameter in this thesis approach was investigated, namely the implementation of vehicles with different characteristics (small, long, normal, and fast) as heterogeneous vehicle groups compared to the normal standard as homogeneous groups of one vehicle type only.

Distribution and Performance Experience with other approaches including multi-agent systems indicates that those models react very sensitively to parameter changes on the level of the agents. This signifies a change in behavior of a category or value, and also implies changes in many other places. Especially the amount of agents which execute the changed behavior or interact with it influences the global system quite significantly. Therefore, the calibration of a model is difficult. Best practices are, first, the use of knowledge of the coherence of the system within the model in order to recognize deficit runs or gain evidence of the parameter changes. Second, an automatized parameter optimization is obtained through learning agents, but not within this thesis.

In order to have sustainable statements about the group process models, a lot more simulation experiments need to run with the same parameter configuration than with a complex but deterministic model. A good model can reduce calculation and storage performance problems for an individual-based simulation.

The goal of the experiments is to have several thousand agents in a rush-hour setting limited in time by one realistic hour, which can take several simulation hours.

The agent behavior is fixed for experiments. A little variation to the agents is to have homogeneous and heterogeneous properties for grouping vehicles as described in Section 6.3. Performance issues were demonstrated in different descriptions of the same agent, see Section 6.3.3.

In this thesis the technical performance is of interest in order to measure the effectiveness and viability of the vehicle group processes. This also depends on three technical indicators: the model, the interpreter and the hardware, which are specified in the following sections.

7.3 Evaluation Scenarios and Settings

This thesis aims to use real-world applications, using a minimum of assumptions and using novel techniques from the area of autonomic agents to contribute dynamic and decentralized models to Traffic Control and Management.

This section compares and evaluates the scenarios presented in Subsection 6.1 as the constructed use cases of this thesis with default behavior, as well as centralized static and decentralized dynamic models of group formation. All scenarios are tested in a randomized manner with at least 100 simulated vehicles in three different simulation instantiations: *without groups*, *with centralized groups* and *with decentralized groups*. For static centralized vehicle groups, all scenarios are divided into *three predefined phases*: the preliminary phase, the main phase and the post-phase. In order to fulfill the goal to *minimize the weights on the edges*, i.e., travel time, shortest/fastest route, CO_2 emissions and fuel consumption, the Dijkstra routing algorithm was used and is discussed in the PLANETS project [110] in detail. Finally, the scenarios are tested with *homogeneous and heterogeneous autonomous vehicle types*, which includes driving behavior but mainly vehicular characteristics, and a heuristic approach of influences of homogeneous versus heterogeneous vehicles is taken into account.

In the test setting it is assumed that the vehicles of the same class have the same properties of desired speed, acceleration, and deceleration. The configured values are described in Chapter 6 in Section 6.3.1 for homogeneous and in Section 6.3.2 for heterogeneous vehicles. The value function in Equation 5.30 also requires the distribution of weights to the properties and the standard derivation. For homogeneous vehicles there is no difference between the vehicles of the same class, therefore the distribution weights can stay equally distributed and the standard derivation can be set to a very low value, i.e., 0.01, which still allows the vehicles of one class to form groups. Then, the vehicles need to be randomly distributed and generated for the simulation, which is done with the random function (presented in Appendix C.2.1). The configuration of the distribution function of different vehicle classes is set to the German city standard distribution described in Section C.4.1, but could be adjusted for future use.

7.3.1 Experiment Description

The experiment description is divided into the following subsections: scenarios 7.3.1, test environment 7.3.1 and measurements 7.3.1 with the multidimensional scaling (MDS) method.

Scenarios

For testing the autonomous vehicle groups in urban traffic, a step-based simulator MATI was developed to model each vehicle from the individual agent perspective included in a traffic environment.

Thus, an elaborate experimental study has been conducted that adds to the findings of the previous studies. An artificial network with two parallel and five perpendicular streets has been modeled in the traffic simulators AIMSUN and SUMO. A partial representation of Hanover, Germany is shown in Figure 6.7.

The vehicles get a initial map with all regulated intersections including their fixed control plans. Realistic traffic flows are used which were collected in a given region of Hanover during morning rush hour between 7:30 and 8:30 am. There are traffic flows in all directions; the focus is on the left parallel street 'Hildesheimer Strasse'. The autonomous vehicles use the modeled graphs, shown in Figures 6.6, for their decisions.

Standard lanes are 3,2 m wide, and each standard vehicle is 1,8 m wide by 4,3 m long (height 1,5 m), unless they have heterogeneous characteristics as a small vehicle (2,2m length, 0,9m width and 1,5m height) or truck/bus (14m length, 2,6m width and maximum 4m height), whereas normal cars and sports cars have the same measurements as the standard vehicle.

The simulator runs with four main parameters represented by different simulation scenarios. Realistic data from Hanover Südstadt was used and adopted for two artificial (the first and last) and two realistic scenarios with different focuses and sizes, illustrated in Figure **??**:

1. a simple **three lane scenario** traveling in only one direction. The distance is three kilometers long, divided into three equal sections with a traffic light for the preliminary, main and post phase.
2. a **green wave scenario, namely the Hildesheimer Straße** (HI) with four sequential traffic lights in one main direction from South to North including contraflow secondary traffic. The preliminary phase lasts from source to the first traffic light, the main phase encompasses the traffic lights until the last node (Aegidientorplatz) before the sink. From there the post phase lasts until the vehicles leave the network in the sink.

3. a **network** of five perpendicular and two parallel streets containing 11 intersections, namely **Hanover Südstadt** (HS). Contraflow and secondary traffic as well as an alternative main route are given. Similar to Hildesheimer Straße, the network has its preliminary phase at the lower end where the vehicles enter until the first traffic lights and finishes with the post phase from the last intersection until they leave the network.
4. a complex scalable **grid scenario** with probabilities of attempting to spawn vehicles in each direction (independently) at the beginning of each time step. For almost all of the experiments, the probability for each direction was the same and randomized. This scenario will show how scalable the experiments are with the given traffic density.

Figure **??** shows a screen shot of the graphical display. In each cycle of the simulator the following events occur:

1. Predefined probability is used to randomly spawn vehicles in each direction in the simulation network. Once a vehicle is spawned, it is placed at random in one of the lanes and travels in that direction. The vehicle is not spawned in case of conflict such as following another vehicle too closely (i.e., within one second or one meter). In case vehicle a is behind vehicle b and the distance between the back of b and the front of a is velocity $a \cdot t$, then vehicle a is considered within t seconds of vehicle b.
2. Each autonomous vehicle (AV) is given the distance to the vehicle in front of it. On-board sensors such as cameras can collect such information and are used to avoid collisions with vehicles traveling in the same direction.
3. Each AV then takes an action based on this information: ACCELERATE (increase velocity by $3m/s^2$), DECELERATE (decrease velocity by $15m/s^2$), or maintain current velocity. The simulator enforces the invariant that $0 \leq speed \leq desiredspeed \leq topspeed$.
4. Depending on the actions the agent takes, an update is made for each vehicle's position and velocity.
5. Any vehicles which finished their journey and have left the network are removed from the simulation is updated. Individual data such as its own delay are tracked by each vehicle. This information is used bottom-up to update the global value for total delay and maximum delay.

The behavior of the AV is completely independent of the traffic environment simulator, because it is encapsulated in the agent interpreter. For the experiments, the same AV type in each vehicle is provided with the group formation protocol. Only vehicle characteristics change for heterogeneous vehicle types. The agent interpreter receives vehicle information from the traffic environment simulator and decides what action to take:

- Maintain speed;
- If $speed < speedlimit$, then ACCELERATE;
- If less than one second or one meter behind the vehicle in front and $speed > 0$, DECELERATE;

- If not through the intersection communication already provided, interact with the group members as specified separately for each group member detailed in Section 6.4.

There are several vehicle types in the heterogeneous execution (see Chapter 6.3.2) with four common typologies: small, i.e., Mini, long, i.e., MAN bus or truck, normal, i.e., VW Golf, and fast, i.e., Aston Martin . The 100 cars have to be split up into these car types. In order to have an official representative statistic, the statistics from the *Kraftfahrt-Bundesamt*[213] were used.

- fast (Aston Martin): 2.974
- normal (VW Golf GTI): 2.064
- small (Smart): 210.728
- long (MAN Bus): 14.533
- Total: 230.299
- Total vehicles in Germany(01/2013): 43.431.124

Now the distribution for the 100 vehicles can be derived:

- fast (Aston Martin): 1
- normal (VW Golf GTI): 1
- small (Smart): 92
- long (MAN Bus): 6
- Total: 100

Test environment

The scenarios described in Chapter 6 are simulated with different parameter values of (1) simulation steps and (2) statistically significant runs of the same scenario with randomized initial vehicle distribution. The parameter of how many vehicles can be in the scenario is fixed at the maximal number of 5275 agents, due to the level of service (LOS) simulating the rush hour traffic derived from the HS location. The experimental scenarios were executed with homogeneous (all the same vehicle characteristics) and heterogeneous (four different vehicle characteristics) vehicle types and with dynamic group formation. The sequential numbering helps to identify the scenarios with their parameter settings in order to compare and discuss in the following.

For each experimental scenario three tables, distables, groupingtables and paramstables, are stored for evaluating the vehicle groups.

- *distables:* the characteristics which are elements in the (metric) (vector) space, i.e., the space to represent speed, a space to represent routes, etc. between the individual vehicles are given, while the similarities of the vehicles are sought.
- *groupingtables:* this table states the number of vehicles and which group they are in.

- *paramstables:* the similarities of the vehicle properties between the individual vehicles are given and the groups of vehicles are sought.

In the following, the alternatives homogeneous versus heterogeneous vehicle groups with dynamic group formation are interpreted and compared with the two realistic scenario locations (HI and HS).

The following 14 characteristics are measured whereas the alphas α belong to the dynamic group formation, which are evenly distributed, but could be varied to give some parameters more impact:

1. $\alpha 1$: max speed (Euclid)
2. $\alpha 2$: acceleration (Euclid)
3. $\alpha 3$: deceleration (Euclid)
4. $\alpha 4$: position (Euclid)
5. $\alpha 5$: destination (Euclid)
6. $\alpha 6$: route (Frobenius)
7. current position x
8. current position y
9. position of destination x
10. position of destination y
11. max speed
12. acceleration
13. deceleration
14. current speed

This results in big data tables with more dimensional data.

Measurements

The *Principal Component Analysis* is a common method to reduce the dimension for linear data. Finding the principal components means finding the underlying structure of the data. That means that the principal components are the directions where there is the most variance in the data, i.e. where the data is most spread out. With the help of *Eigenvectors* and *Eigenvalues*, the principal components can be calculated. Eigenvectors and Eigenvalues always come in pairs, i.e. every Eigenvector has a corresponding Eigenvalue. The amount of Eigenvectors and/or -values equals the number of dimensions of the data set. Global non-linear methods are MDS, Isomap, MVU, Kernel PCA, Diffusion Maps, and Multilayer Autoencoders. For more information on the techniques, see the following publications: `Maaten et al.` [219] and `Maaten and Hinton` [358].

Some are implemented in the drtoolbox made for the data analysis of this thesis (see Appendix C). Most of these techniques simply provide tools to display more than two data dimensions, but the interpretation of the data requires a human observer. Thus, the applicability of these techniques is severely limited for real-world data sets that contain thousands of high-dimensional data points. In many different domains the visualization of high-dimensional data remains an issue, because data is widely varying in dimensions. Vehicle characteristics that are relevant to urban traffic, and for this thesis, are described by fourteen variables.

Dimensionality reduction methods preserve the significant structure of the high-dimensional data in the low-dimensional maps. The reduction alleviate the problematic visualization techniques by converting the high-dimensional data set $X = \{x_1; x_2; \dots; x_n\}$ into two or three-dimensional data $Y = \{y_1; y_2; \dots; y_n\}$. Therefore, they can be used to display scatter plots for highly-dimensional data, which is used for the purpose of this thesis. Traditional dimensionality reduction techniques are linear techniques that focus on keeping the low-dimensional representations of dissimilar data points far apart such as the Principal Components Analysis (PCA; [170]) and classical multidimensional scaling (MDS; [347]). Classical multidimensional scaling (MDS) gives the best results for the evaluation and is included in the MATLAB script.

Classical scaling of the MDS (cf. [358]) finds a linear transformation of the data, minimizing the sum of the squared errors between high-dimensional pairwise distances and their low-dimensional representatives. MDS as a linear method with classical scaling focuses on preserving the distances between widely separated data points, but its weakness is in preserving the distances between close data points. The MDS procedure only makes it possible to judge the similarities without the direct relation of concrete characteristics and tries to configure/adjust the objects in a latent room in such a way that the similarity results are best reproduced. Therefore, the vehicle objects are arranged in a room of latent dimensions.

Given is a set of observations of the simulation about the sensory distances of the objects which are saved in matrices of the distances. The problem which MDS solves is to find the latent variables, so that the distances between the vehicle objects can be represented on the coordinates. In this thesis, the pairwise similarities/dissimilarities of vehicle characteristics (maximum speed, acceleration, deceleration, position, destination, and route) are evaluated. In order to do so, the MDS provides the latent perception mapping on which the vehicles are arranged, so that the similarity results are reproduced.

The procedure of MDS is

1. measure the similarities: ranking, fixture point, or rating
2. select a distance model: Minkowski metrics $p = 2$, $p = 1$ or $p = ¥$
3. determine the configuration: if applicable, rotate axis, mirroring
4. number and interpretation of the dimensions: here six dimensions
5. aggregation of vehicles

The first point of MDS consists of three methods:

- **ranking method:** 'n of 2' pairs are ordered from the most dissimilar to the most similar pair, which is hardly possible with big values of n. The number of vehicles n is limited to 5275.

- **fix point method:** Every vehicle is the comparison object (fix point) for all other vehicles once. Depending on the number of vehicles (maximum 5275), the same rank rows are constructed. Asymmetric quadratic distance matrices are obtained, which can be converted to symmetrical ones.
- **rating method:** All possible pairs are generated, then evaluated on a rating scale depending on the similarity and presented in a randomized manner.

Problems can be caused by the ties and the reliability of ranks.

The second point of the metrics needs to be chosen according to content, because the distances are selected appropriately. There are three metrics called the Minkowski metrics:

- $p = 2$
  Euclidean metric is the distance between vehicles represented by the length of the connecting line
- $p = 1$
  City-Block metric is the distance between vehicles which is the sum of the individual coordinate distances, or
- $p = ¥$
  Supremal metric is the distance between vehicles which is the largest of the resulting coordinate distances.

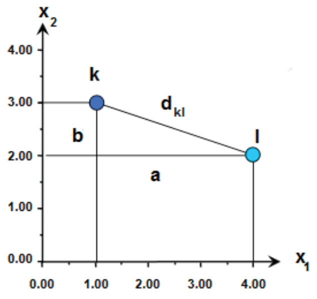


Figure 7.1: Euclidean Metric.

Figure 7.1 with the Euclidean Metric sets the following equations

$$\begin{aligned} a &= x_{k1} - x_{l1} &&= 1 - 4 = -3 \\ b &= x_{k2} - x_{l2} &&= 3 - 2 = 1 \\ d_{kl} &= \sqrt{3^2 + 1^2} &&= \sqrt{10} = 3.16 \end{aligned} \tag{7.1}$$

in which the last row describes the object distance.

After that, in the fourth step, the configuration can be determined. Starting with the dissimilarities u, the task is to find a low dimensional space, where the distances d comply with the monotonous relationship if $u_{kl} > u_{ij}$ then $d_{kl} > d_{ij}$.

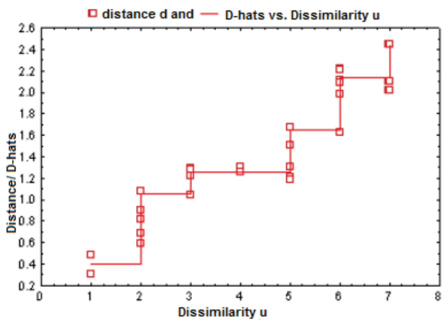


Figure 7.2: Shepard Diagram.

The Shepard diagram in Figure 7.2 shows dissimilarities and their monotonous relationship.

In order to determine the configuration, a table of dissimilarities between objects can be made, as well as a table of the coordinates of $x1$ and $x2$ and the objects (vehicles) with their Euclidean distances in relation to each other, the distance $d(kl)$, the rank of $d(kl)$ and the dissimilarities ukl. Then the Shepard diagram with the dots of dissimilarities ukl on the ordinate and the rank of $d(kl)$ can be plotted as a starting configuration which will adjust to a line with new coordinates disparity which deviates and can be expressed by the quality criterion STRESS (STandardized REsidual Sum of Squares)

$$Stress = \sqrt{\frac{e_{kl(d_{kl}d_kl)}^2}{Factor}}$$

.

The more dimensions each object has, the smaller the Stress value is, where solutions with a low amount of dimensions are easier to interpret. Stress is not allowed to be zero, which would lead to an unclear solution. The trade-off of low stress value versus the interpretation with a high Q-value is reached by adding more dimensions in order to reach more representation. At the same time, the data agglomeration strives against the adding of the amount of objects n (vehicles) but results in a better concentration, but worse interpretation of precision.

Rules for orientation in interpretation may rotate the axis, stress should be low and the data should have an agglomeration degree Q, where Q is as big as possible in the table. The

$$Q = \frac{\text{amount of similarities}}{\text{amount of coordinates}}$$

.

The fifth step represents the aggregation of vehicles in the MDS outcome.

The MDS as a classical method serves to determine the configuration of one vehicle. Only in the last step are techniques for aggregations executed pertaining to similarity data, configurations, and with special calculation methods for MDS analyses on the similarity data of several vehicles with replicated MDS.

The advantages of the MDS method are that relevant characteristics can be unknown, there is no falsification through previous selection. It can also be used with priority data order, because the results are nearly identical with metric MDS. For these investigations this was not relevant. One main disadvantage can be the aggregation of vehicles, which is problematic in reference to different characteristics which are included in the result. The MDS is not algorithmic, that means without a guarantee of the best solution, and the solution is dependent on additional parameters like the distance model and the amount of dimensions.

For this thesis, those disadvantages were not taken into account, because the α-values are used with the same weight within the adapted Euclidean distance model, no best solution is required and the distance model includes six dimensions logged in the 'distables', whereas the 14 vehicle characteristics are listed in the 'paramstables' and the 'groupingtables' includes the similarities or dissimilarities of which vehicle belongs to what group, so that in visualization they can be represented with different colors.

The MDS is an exploratory method. Therefore, no strict hypotheses can be evaluated. For the purpose of this thesis it is sufficient to show the relative distances between the vehicles. Known problems with the configuration (relative positions of the objects within the perception mapping, if just the distances are known), determination of the dimensions and the determination of the metrics are handled for the interpretation of vehicles' dissimilarities. The MDS configuration is independent of rotation and mirroring, which effects the interpretation.

7.3.2 Experiment Execution

The data is collected in each simulation by MATI with 'distables' to log the dissimilarities value of the Adapted Euclidian distance function for vehicle grouping algorithm (AEDF) in each step for every vehicle pair. This table is a matrix with the same values mirrored along the 0-value-diagonal. In 'grouping tables', all the vehicles are logged in each time step with their grouping value. Therefore, groups can be identified. The 'paramstables' logs the 14 different values included in the AEDF function. Hence, every influencing parameter of the vehicle can be checked individually. All the tables are filled with the maximum number of vehicles, 5275. That also means that vehicles not in the simulation are filled with the '0' value, which makes the files quite big. Data is logged for runs of 1000 time steps, because it is easier to scale for statistical evaluation. The same scenario can run several times in a row with random start configurations in order to measure the statistical impact.

The vehicle groups are assessed by their mean speed, their travel time and the delay of a vehicle class. Simulation of Urban MObility (SUMO) calculates the speed of a vehicle.

Table 7.1: Model Calibration.

Category	Parameter	Source	Adjustment
Network	street geometry	city map	added and simplified
	street types	observe	added and changed
	v_{max}	observe	added and changed
	lanes	count	added
	traffic lights	TUBS	signal plans
Demand	traffic demand	count	calibration
	flow data	detector	calibration
	travel time	SUMO	-
Driver/Vehicle	behavior	MATI	added/calibration
	O-D information	detector	with changes
	traveler data	knowledge	-

According to SUMO wiki, a vehicle in SUMO consists of three parts: a description of vehicle's physical properties summarized by the vehicle type, a pre-defined route as the vehicle driving plan, and the physical vehicle in simulation. The attributes speedFactor and speedDev are used to sample a vehicle specific chosenSpeedFactor from a normal distribution with mean speedFactor and deviation speedDev. Using speedFactor=1, speedDev=0.1 will result in a speed distribution where 95% of the vehicles drive between 80% and 120% of the legal speed limit. For flows, every inserted vehicle will draw an individual chosenSpeedMultiplier. In order to prevent extreme idling the resulting values are capped at the low end by 20% of speedFactor. A vehicle keeps its chosenSpeedMultiplier for the whole simulation and computes the actual speed for driving multiplying the street speeds. Thus, vehicles can exceed edge speeds. However, at the vehicle type's maxSpeed the vehicle speeds are capped. When the departSpeed of a vehicle exceeds the speed limit and its vType has a speedDev > 0, it can accommodate the stated departure speed, because the individual chosenSpeedMultiplier is high enough.

As an initial test of the MATI simulator, the theoretical predictions are verified with matching empirical performance for the scenarios that can be analyzed completely. Some simulations were executed to determine both the average and maximum delays as functions of both and P.

Model Setup

In order to use the simulation model as a realistic prediction, it needs to be adjusted to the conditions of the simulated environment and time (time of day or day of the week). The model calibration occurs in different categories [165].

For calibrating the simulation, the parameter can be adjusted as illustrated in table 7.1 for a rough overview. In a sophisticated scenario study, every item can be refined and differentiated, but here, the initial calibration results from the study purpose 'vehicle groups' and field data from the Southern part of

Hanover, Germany during rush hour. Additionally, variances of average values can be of importance:

- average speed on a street segment *retrieved from simulation data
- mean travel time between two points *retrieved from simulation data
- average traffic flow at a measuring point like a detector *no data available
- mean traffic density on a section *retrieved from measured real data
- average fuel consumption or pollutant emission *retrieved from simulation data
- number of accidents *no data available

In traffic simulation, calibration processes are difficult due to little measuring data (available). Expert knowledge is necessary to decide whether deviations between measured values and simulation results can be accepted (cf. [165]). The different categories are detailed in the following:

In general, traffic models of a network are abstracted in order to be represented in a simulation. They consist of links which are street segments and nodes which are the intersections of the streets. Additionally, there are zones which are sources and sinks for vehicles to enter and leave the simulation. Network spatial extents and the level of detail in the model need to be decided on. In the category of the network (in particular the Southern part of Hanover, Germany) the street geometry from the city map is used. It is added into the traffic simulator (i.e. AIMSUN or SUMO) and simplified. Street types and maximum velocity are observed and simplified: minor streets are 'upgraded' to normal priority higher speeds (50km/h) so that re-routing in the network becomes more feasible in the small scenario extract of a city. Re-corrections are required if more realistic traffic should be predicted. OpenStreetMap[1] (OSM) data can be integrated into the traffic simulator or a network file can be constructed manually. Extensions or corrections need to be done if the OSM data is incomplete or incorrect. Often the positions of traffic lights or speeds are missing.

`Hellinga` (cf. [165] p. 20) defines the Origin-Destination (O-D) demands by the amount of trip demands between each origin area and each destination area during a particular departure time period. Usually, traffic simulation models only represent vehicle traffic and not individual trip demands. This is another practical reason why the focus is just on autonomous vehicles where the demand was counted for the Southern part of Hanover, Germany (HS). Dynamic O-D demands are required and the most dynamic demand is a list of each vehicles individual departure time and usually in the range of 15 minutes to one hour.

Typically, microscopic traffic simulation models require the specification of macroscopic flow characteristics or the observed data is obtained from loop detectors and aggregated to a 5-minute average. For the model representation of the Southern part of Hanover, Germany, the latter is used.

[1]http://www.openstreetmap.org

Autonomous vehicle behavior is important for the routing choice considered in the studies [106] [137] and [107] in which the modeled network presents AVs with more than one viable route choice. The accurate representation of a vehicle's route choices is essential. Driving behavior is abstracted to agents, all of the same type. Vehicle characteristics are taken into simulation additionally.

Traffic systems are calibrated for the driving behavior in the area of the street graph or network file, e.g., routing is specified. A simple method is to select residential areas as the origin and industrial areas as the destination in the morning, which would result in rush-hour traffic. This simple method gives the general idea. The exact routes are identified by origin-destination matrices (ODM_{ij}), where every source area i is connected with the target area j and allocates the amount of AVs which move from i to j. The determination of ODM can be acquired by people census, traffic counts or aggregation of detector data. The time of day can be simulated by ODM^t_{ij} where the amount of traffic participants is allocated which move from i to j at the point in time t. Usually, aggregated data is used for time intervals. Partially, an additional value is the traffic participant type.

The calibration with ODM is required for a concrete scenario in order to achieve a realistic volume of traffic. In order to calibrate MATI, the microscopic traffic models in Section 2.2.4 of car following and lane changing were investigated.The calibration process is mainly provided by the traffic simulator itself and the effects of influencing actions of routing and especially grouping were investigated.

An idea is to reduce the average fuel consumption and, at the same time, increase the traffic flow, which might be possible through smart vehicle group behavior as discussed in the results 7.5.1.

Simulation Time

The simulations run on a standard computer, the simulation time is stated for the realistic scenarios with different runs, but the same 1000 simulation steps.

Hildesheimer (HI)

- 3 runs, 1000 steps, 1 day and 2 hours
- 5 runs, 1000 steps, 2 days
- 10 runs, 1000 steps, 4 days and 3 hours
- 15 runs, 1000 steps, 6 days and 6 hours

Südstadt (HS)

- 3 runs, 1000 steps, 4 days
- 5 runs, 1000 steps, 4 days and 3 hours
- 15 runs, 1000 steps, 12 days and 12 hours

Justification: Using TraCI slows down the simulation speed which depends on many factors:

- per simulation step the number of TraCI function calls,
- types (some being more expensive than others) of called TraCI functions,
- the TraCI script computation within, and
- the English client language.

7.3.3 Data Analysis

In order to get a consistent format, the output data for MATI scenarios are described in the Appendix C.4. Additionally, a consistent format is needed in order to evaluate the data inside the MATLAB platform.

The Hierarchical Data Format (HDF), and particularly HDF5, a unique technology suite that makes the management of extremely large and complex data possible, is used as the data format.

7.4 Simulation Results

In this section, the results attained during the grouping phase, based upon the solution concept introduced and projected on selected scenario configurations, are described and discussed.

The effects of dynamic vehicle groups as well as cooperative behavior are of special interest for this thesis. Furthermore, more detailed information on traffic flow for specific traffic streams at intersections was needed. The main research question is answered through decentralized autonomous vehicle groups with interaction and the initiation of cooperative traffic behavior supported by the infrastructure which includes:

- the representation of traffic participants as autonomous interacting agents
- individual versus global decision making especially for grouping vehicles according to their ODM
- the reduction of travel times with the strategy of grouping

In this section, the performance of group formation is evaluated against the dynamic vehicle groups for stable amounts of traffic (LOS D), variations of network scenarios, and granularities of the groups.

First, it is important to note that the amount of traffic that each scenario handles is the same, adopted from the reference of Hanover Südstadt (HS) in the morning rush-hour of 7:30 to 8:30 a.m. The simulation system can handle as much traffic as is generated. It is interesting, though, to examine the throughput of the urban traffic system as a default scenario without groups versus scenarios with vehicle groups. The fixed reference is the scenario time of 3600 seconds to simulate the morning rush-hour. The variables are the actual amount of vehicles passing through the network. For example, the HI Green Wave scenario has 2723 vehicles passing through the network without groups, which is a bit more than half of the reference scenario of HS with 5275 vehicles during the same hour.

Example HI 'tripinfo.xml' output:

- total number of vehicles: 2723.0
- total departure delay(s): -115.5,
- average departure delay(s): -0.0424164524422
- total waiting time(s): 1289296.5,
- average vehicular waiting time(s): 473.483841351
- total travel time(s): 328926.0,
- average vehicular travel time(s): 120.795446199
- total travel length(m): 2811778.28,
- average vehicular travel length(m): 1032.60311421
- average vehicular travel speed(m/s): 9.38178901666

One problem experienced was how to make the scenarios comparable. This is solved with the reference scenario, which is good for smaller scenarios, but is limited for the grid scenario, which should give insights into how scalable and general the conclusions are.

Qualitatively, the group formation is able to sustain a much higher throughput than any of the traffic signal systems before causing the network to be used more efficiently.

Secondly, and more precisely, the analyzed parameters include speed and position data such as travel route, travel time, current speed and stop time, which are compared.

The whole system in the form of travel time, throughput, and delay has been executed separately: the same location in a different simulation called AIMSUN, described in Subsection 7.4.1.

For the evaluation, in MATI using the SUMO simulator, 161 experimental trails were run with homogeneous and heterogeneous scenarios as well as two realistic locations, namely, Hildesheimer Street as a part of the Southern part of Hanover. The averages of the following metrics distance, group formation and parameters influencing the groups were plotted for statistical analysis.

7.4.1 Analysis of Group Efficiency

To investigate the influence of parameters on the effectiveness of the group formation, the input parameter is the ratio of equipped vehicles to the total number of vehicles. For example, the penetration rate illustrated in the combined Figure 7.5, which was published in `Fiosins et al.` [110]. Those experiments were done in the AIMSUN traffic simulator and were the starting point for grouping vehicles in urban traffic. A role-based grouping method as a version of the contract net protocol was used for the Hildesheimer Street (see Appendix D).

Throughput and Delay

A primary concern, however, is how efficient the autonomous vehicle groups are. This relates to the first set of research questions and hypotheses 7.1.1. To measure vehicle group efficiency, the specific metrics related to throughput and delay are considered. Safety is considered a prerequisite throughout this evaluation: when the autonomous vehicles follow the protocols correctly 0 probability of collisions are assumed for all control policies.

Throughput is one metric which measures the amount of traffic that can be handled in a simulation scenario. While this is hard to measure precisely, qualitative claims are made regarding the throughput of several different scenario systems. The throughput is the rate of successful vehicles passing through the urban traffic system within a defined network. The throughput of a traffic system may be affected by various factors, including the limitations of the underlying analogous physical medium, available processing power of the system components, and end-user behavior. Average throughput considers the set of all vehicles passing through a scenario in a period of time, also known as traffic flow.

Definition 7.1 (Throughput) *Let ΔN be the average of fixed vehicle numbers in a simulation (this thesis has a maximum of 5275 vehicle agents within a simulation) and let Δt be the average time. Then the traffic flow is defined as:*

$$Q(x,t) := \frac{\Delta N}{\Delta t} = \frac{max\ 5275 vehicles}{average\ time}$$

Delay is considered as the primary metric; what effect does the presence of autonomous vehicle groups have on the overall journey of such a vehicle? The differentiation is to focus on the average delay, but also to identify the worst case.

Definition 7.2 (Delay) *Let C be the period of time in which all vehicles pass through a scenario. Assuming that no other vehicles were on the street, the trip from point A to point B takes vehicle v_i $t_0(i)$ time. However, due to other vehicles in the same lane and due to having to go through intersections involving other vehicles, v_i arrives at time $t(i)$ instead. Then the average delay of an intersection is defined as:*

$$\frac{1}{|C|} \sum_{v_i \in C} t(i) - t_0(i)$$

The worst case delay is defined as the maximum delay:

$$\max_{v_i \in C} t(i) - t_0(i)$$

Given these metrics, the theoretical performance of some scenario systems can be analyzed; for example, a mini intersection scenario or multiple sequential intersections in a green wave scenario, which is equivalent to the realistic scenario of Hildesheimer Street. Analysis including the traffic light model is somewhat tricky. In this model, a vehicle has a random chance of making it through the intersection unimpeded. Fixed signal plans are used in order to make calculation of the rest time phases more predictable for the vehicle agents. In the case of approaching a red signal phase, the autonomous vehicle must come to a complete stop and wait.

The average and maximum delays are complicated functions of the timing of the signal lights (how long they are red, yellow and green), how many vehicles are on the street, and what the velocities of the others are, among other things. These are the independent variables. In order to analyze this model of the green wave scenario 1.2, some simplifying assumptions are made.

1. *Vehicles traveling in the same direction interact with one another, but not opposing traffic.* Vehicle-to-Vehicle communication, same as C2C (V2V) interactions are incorporated in the model, but limited to broadcasting and multi-hop forwarding in the direction from origin to destination. For a theoretical lower bound V2V, interactions can only increase the delay for the average vehicle, which is not considered in the analysis, but discussed in Section 5.5.
2. *Vehicles that have to decelerate due to a red light reach full speed shortly after the light turns green again.* Again, the lower bound of traffic can be modeled to give the autonomous vehicle more information than a normal traffic light would, namely to transmit the rest time value of the current signal phase. This allows the autonomous vehicle to accelerate at just the right time, such that it loses the minimum amount of time waiting at the traffic light.

Because the vehicles are assumed to be grouped, the average delay per vehicle can be found in the FCD (cf. [96]).

Consider the following parameters:

P as the traffic light phase

α as the green phase for one direction.

From the definitions of these parameters, two constraints can be specified: $P > 0$ and $0 < \alpha \leq 1$

Assuming that the vehicles always reach the green light at top speed, any dependence on the acceleration or deceleration of the vehicle can be removed. For calculating the delay of one vehicle, the difference is calculated between the idealized vehicles arrival time if it were always green and the expected vehicles arrival at the traffic light.

In the end, the answer is dependent only on the green time period α and the period of the traffic light P. Without deceleration and no delay α of the time, the vehicle reaches the green light. The remaining $(1 - \alpha)$ of the time, the vehicle does not reach the intersection until the beginning of the next cycle. Since the red time phase is $(1 - \alpha)P$, the average waiting time for a vehicle will be $\dfrac{1}{2}(1 - \alpha)P$. Thus, the total expected delay for a vehicle is $\frac{1}{2}(1 - \alpha)^2 P$.

The expected delay approaches zero when α approaches 1. And vice versa when α approaches zero, the expected delay reaches $\dfrac{1}{2}P$. This is slightly counter-intuitive, because common sense dictates that the delay should grow without limits. But it is important to remember that, the instant the signal turns green, the vehicle is allowed to drive. When the light phase increases and vehicles

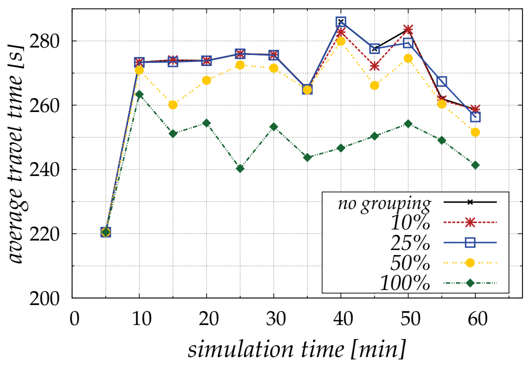


Figure 7.3: Parameter Travel Time.

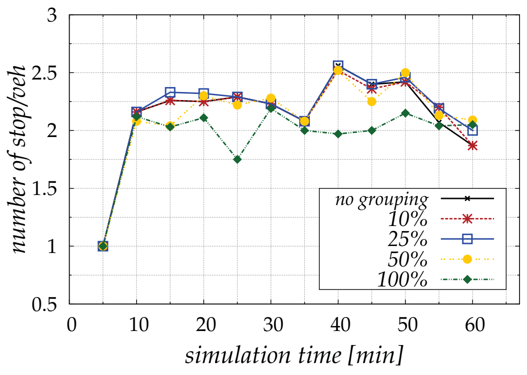


Figure 7.4: Parameter Stop Time.

Figure 7.5: Grouping Effects measured with different Penetration Rates including Throughput and Delay from [109] p. 14.

missing the green wave, the expected delay joins and increases linearly because the have to wait longer.

The maximum delay can be calculated similarly. An autonomous vehicle must wait the longest amount of time when it is red, or $(1 - \alpha)P$.

Travel Time and Stop Times

For statistical validity of the results, each data point illustrates the average of at least 10 simulation runs with random initial sets and one hour simulated time per run for all figures in this section. In order to be able to assess the impact of the grouping performance on traffic efficiency, the focus is on the mean travel time (Figure 7.3 correlating with travel time) as well as the number of stops per vehicle (Figure 7.4 visualized with stop time) along Hildesheimer Street. Figure 7.5 presents the aforementioned traffic efficiency metrics as a function of the simulation time for different penetration rates. With a penetration rate smaller than 50%, the mean travel time is reduced by up to 14% and the number of stops per vehicle can be reduced by up to 20%. The simulation results obtained for different penetration rates clearly show that the ratio of equipped vehicles has a significant influence on traffic efficiency.

Figure 7.3 correlates the travel time plots with the throughput that, on average, each autonomous vehicle achieves after every complete run in the Hildesheimer scenario. After 10 simulation runs, the throughput evens out at around 240 seconds after one hour of simulation for the Hildesheimer network for fully equipped autonomous vehicles. Without grouping and at penetration rates of 10% and25% the results do not differ much and close around 260 seconds after one hour of simulation. The half-equipped vehicles have a throughput of 250 seconds and represent, as expected, the middle range compared to no grouping and fully equipped vehicles. It is worth pointing out that the distribution over time shows a peak after 10 minutes at 260 to 275 seconds travel time for all penetration rates , but the general progression is lower, namely, around 250

Table 7.2: Array Distances.

	GroupCount	$\overline{mean\_dist}$	$\overline{sd\_dist}$	$\overline{scenario}$	$\overline{location}$
All	4000	11.864	64.079	0.5	0.5

seconds, at all times for the 100% grouping compared to the similar curves for the other penetration rates which are at around 275 seconds.

Figure 7.4 is visualized with stop time, thus, plots the delay. It is appreciable how each autonomous vehicle at all penetration rates approaches a stop time close to the lowest possible delay with 2 stops after 10 iterations. The curve of 100% penetration is, once again, lower, and ranges around 2 stops at all times with a low peak of 1.75 stops at 25 minutes. On the other hand, the other progressions of the curves are quite identical with the 50% line a bit lower than the others, as expected, ranging around 2.25 stops with a high peak of 2.5 stops at 40 minutes.

In Chapter 4.3.4, the ATSim simulator was also tested with the GoD method on an artificial three lane scenario. It proved that autonomous vehicle groups were more beneficial than individual driving, as represented in Figure 4.12. Thus, similar results are expected with the MATI simulator at the realistic scenario location of Hildesheimer Street and Hanover Südstadt.

In the following sections, the similarities 7.4.2, group formation 7.4.3 and the parameters 7.4.4 are analyzed, giving answers to the research questions and hypotheses in Section 7.1.

7.4.2 Statistic Analysis of Distance

In this section, the statistic evaluation of the distance data is conducted. For the elementary analysis, a table object (see Table 7.3) was constructed with the previously generated array (see Table 7.2), which simplifies further analysis (see Appendix C.4). The notation $\overline{abc}$ indicates the mean value.

Output:

The Table 7.2 array distances has a group count of 4000 overall, a global mean of all the mean distances of 11.864, a global mean of all the mean variations of 64.079 and a choice of two scenarios as well as two locations.

The idea is: Imagine a vehicle that, at distance d from its destination, travels at the speed s per second. The natural logarithm of $\overline{x}$ consists of the time in seconds which the vehicle takes in distance (or speed) to be reduced by a factor $\overline{x}$. This is a description in terms of traveling at variable speeds. The overall (i.e., integral) reduction factor is described by the cumulative effect of traveling at variable speeds or distance.

The Table 7.3 relates to the next two plots, Figure 7.6 and Figure 7.7, for their linear regression models.

Two variables are available for the distance.

1. *mean_dist*: mean distance between AVs per step
2. *sd_dist*: mean variation of the distance between AVs per step

Table 7.3: Elementary Table Object Distances.

	scenario	**location**	**GroupCount**	$\overline{\textbf{\textit{mean\_dist}}}$	$\overline{\textbf{\textit{sd\_dist}}}$
0_0 blue	0 (homo)	0 (HI)	751	6.4443	44.721
0_1 purple	0 (homo)	1 (HS)	1000	19.222	90.104
1_0 yellow	1 (hetero)	0 (HI)	1000	4.8186	38.055
1_1 red	1 (hetero)	1 (HS)	1000	17.605	86.091

For each of those variables, a plot is generated which shows the distribution depending on the homogeneous and heterogeneous scenarios and the Hildesheimer and Südstadt locations. Furthermore, a linear regression model is created which models the mean differences between scenario and location as well as the interdependency between scenario and location, expressed in the following equation 7.2:

$$y_i = \alpha + \beta_1 scenario_i + \beta_2 location_i + \beta_3 scenario_i * location_i + \varepsilon_i \tag{7.2}$$

The beta coefficients in Equation 7.2 represent the 'mean effect' of each influencing factor on the depending variable. Additionally, the interdependency of the scenario and the location can be quantified.

Mean Distance

Linear regression model for mean distance:

$$mean\_dist \sim 1 + scenario * location \tag{7.3}$$

The results of the regression model for mean distance (*mean_dist*) (see Appendix C.4.4 show a significant effect of the influencing factors of scenario and location $F = 4.16e + 03$ and $p < 0.001$. The information of the variance of the model is located in a high range with $R2_a dj = 0.757$. In this connection a significant difference is shown between the locations, where the mean distance for the Südstadt location is $b = 13.41$ measurement units higher than for the Hildesheimer location $p < 0.001$.

A significant difference also exists between the homogeneous and heterogeneous scenarios. In the homogeneous scenario, the mean distance between the autonomous vehicles (AVs) is about $b = -0.99$ measurement units lower with $p < 0.001$. Additionally, the effect of the scenarios is dependent on the simulated location. In the Südstadt location, the negative effect of the homogeneous scenario increases by about $b = -0.62$ units with $p < 0.01$.

The mean variation of the distance shows a similar result.

Mean Variation

Linear regression model for mean variation:

$$sd\_dist \sim 1 + scenario * location \tag{7.4}$$

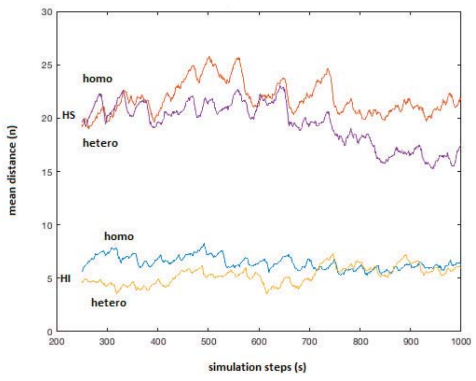


Figure 7.6: Mean Distance.

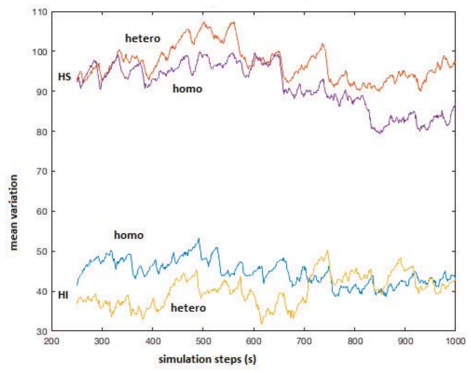


Figure 7.7: Mean Variation.

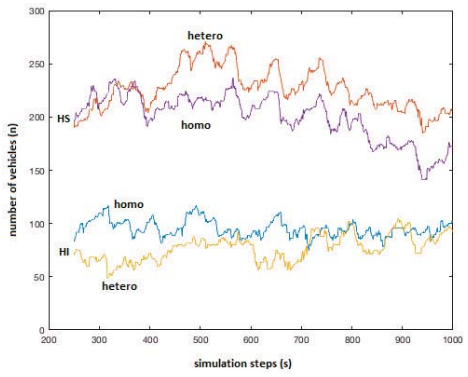


Figure 7.8: Number of Vehicles.

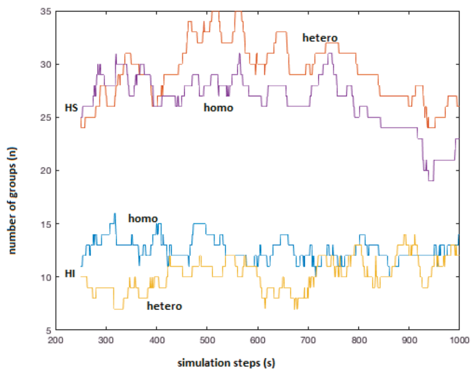


Figure 7.9: Number of Groups.

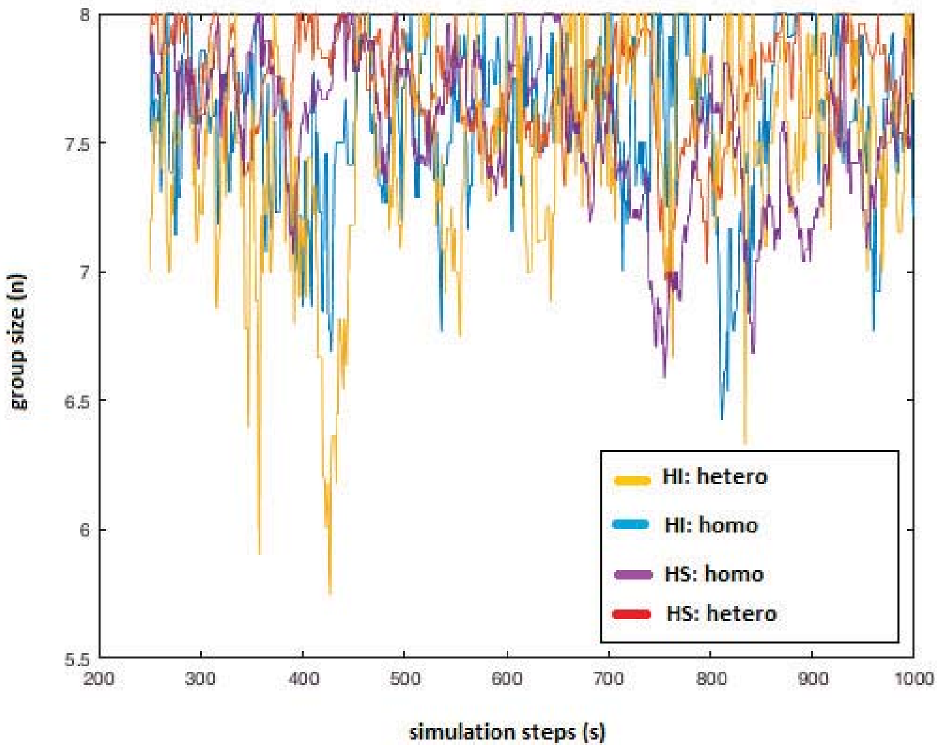


Figure 7.10: Size of Groups.

Table 7.4: Array Group Formation.

	GroupCount	$\overline{n\_veh}$	$\overline{n\_group}$	$\overline{size\_group}$	$\overline{scenario}$	$\overline{location}$
All	4000	133.2	17.53	7.5285	0.5	0.5

Table 7.5: Elementary Table Object Group Formation.

	scenario	location	GroupCount	$\overline{n\_veh}$	$\overline{n\_group}$	$\overline{size\_group}$
0_0 blue	0 (homo)	0 (HI)	1000	85.507	11.307	7.5314
0_1 purple	0 (homo)	1 (HS)	1000	197.47	25.638	7.6707
1_0 yellow	1 (hetero)	0 (HI)	1000	69.984	9.459	7.3566
1_1 red	1 (hetero)	1 (HS)	1000	179.83	23.718	7.554

The entire model of the mean variation of the distance is statistically significant ($F = 5.17e03$, $p < 0.001$). The information of the variance of the model is located in a high range with $R2_adj = 0.795$. The mean variation of the distances is about $b = 48.04$ measurement units higher for the Südstadt location with $p < 0.001$. The homogeneous scenario reduces the mean variation on average by about $b = -4.01$ units. In contrast to the mean distance, the mean variation shows no significant interdependencies between scenario and location with $b = -0.003$.

7.4.3 Statistical Analysis of Group Formation

In this section, the statistical evaluation of the group formation data is conducted, which relates to the research questions and hypotheses 7.1.3. For the elementary analysis, a table object (see Table 7.5) was constructed with the previously generated array (see Table 7.4).

The Table 7.4 array group formation has a group count of 4000 overall; the global mean of all vehicles is 133.2, the global mean of all groups is 17.53 with a global group size of 7.5285 and a choice of two scenarios as well as two locations.

The Table 7.5 relates to the next three plots, Figure 7.8, Figure 7.9 and Figure 7.10, for their linear regression models.

Three variables are available for group formation.

1. *n_veh*: amount of AVs per step
2. *n_group*: amount of AV groups per step
3. *size_group*: number of AVs in a group per step

For each of those variables, a plot is generated which shows the distribution depending on the homogeneous and heterogeneous scenarios and the Hildesheimer and Südstadt locations. Furthermore, a linear regression model is created which models the mean differences between scenarios and locations as well as the interdependency between scenario and location expressed in equation 7.2, described above 7.4.2.

Number of Vehicles

Linear regression model for group formation - amount of vehicles:

$$n\_veh \sim 1 + scenario * location \tag{7.5}$$

Number of Groups

Linear regression model for group formation - amount of groups:

$$n\_group \sim 1 + scenario * location \tag{7.6}$$

Group size

Linear regression model for group formation - group size:

$$size\_group \sim 1 + scenario * location \tag{7.7}$$

7.4.4 Statistical Analysis Parameter

The Sidewaystable 7.6 presents the extensive array parameter with a group count of all over 4000. The global mean of each alpha value is 0.16533, the global mean of position x is 629.06 and position y is 1039.3, the global mean of destination x is 571.41 and destination y is 1109.2, the global mean of maximum speed is 45.343 and the global mean of current speed is 7.6797 whereas maximum acceleration is 2.3657 and maximum deceleration is 5.2277 with a global sum of stop times of 60262 and a global average stop time of 0.98658 and a choice of two scenarios as well as two locations.

The big Table 7.7 relates to the following twelve chosen plots: the same plot of the six alpha values to the other six parameters (details are in the Appendix C.4.6), which are actual position (Figures C.9 and C.10), destination (Figures C.11 and C.12), speed (Figure 7.14), acceleration and deceleration (Figure 7.12), and stop times (Figures 7.15 and 7.16) for their linear regression models.

Alpha 1 to 6

The α values all have the same values - the unified distance parameter resulting from the Adapted Euclidean Distance Function. $x = [\alpha 1, \alpha 2, ..., \alpha 6]$

Linear regression model for parameter - $\alpha 1$ to $\alpha 6$:

$$alpha\_x \sim 1 + scenario * location \tag{7.8}$$

The alpha values are almost steady in the absolute numbers for the scenarios and locations , therefore they are plotted for demonstration purposes in the Appendix C.4.6.

Table 7.6: Array Parameter.

	GroupCount	$\overline{\boldsymbol{\alpha\_1}}$	$\overline{\boldsymbol{\alpha\_2}}$	$\overline{\boldsymbol{\alpha\_3}}$	$\overline{\boldsymbol{\alpha\_4}}$	$\overline{\boldsymbol{\alpha\_5}}$	$\overline{\boldsymbol{\alpha\_6}}$
All	4000	0.16533	0.16533	0.16533	0.16533	0.16533	0.16533
	$\overline{\boldsymbol{pos\_x}}$	$\overline{\boldsymbol{pos\_y}}$	$\overline{\boldsymbol{dest\_x}}$	$\overline{\boldsymbol{dest\_y}}$	$\overline{\boldsymbol{max\_speed}}$	$\overline{\boldsymbol{max\_accel}}$	$\overline{\boldsymbol{max\_decel}}$
All	629.06	1039.3	571.41	1109.2	45.343	2.3657	5.2277
	$\overline{\boldsymbol{cur\_speed}}$	$\overline{\boldsymbol{sum\_stand}}$	$\overline{\boldsymbol{mean\_stand}}$	$\overline{\boldsymbol{scenario}}$	$\overline{\boldsymbol{location}}$		
All	7.6797	60262	0.98658	0.5	0.5		

Table 7.7: Elementary Table Object Parameter.

	scenario	**location**	**GroupCount**	$\overline{\alpha\_1}$	$\overline{\alpha\_2}$	$\overline{\alpha\_3}$	$\overline{\alpha\_4}$
0_0 blue	0 (homo)	0 (HI)	1000	0.16517	0.16517	0.16517	0.16517
0_1 purple	0 (homo)	1 (HS)	1000	0.16583	0.16583	0.16583	0.16583
1_0 yellow	1 (hetero)	0 (HI)	1000	0.16467	0.16467	0.16467	0.16467
1_1 red	1 (hetero)	1 (HS)	1000	0.16567	0.16567	0.16567	0.16567
	$\overline{\alpha\_5}$	$\overline{\alpha\_6}$	$\overline{pos\_x}$	$\overline{pos\_y}$	$\overline{dest\_x}$	$\overline{dest\_y}$	$\overline{max\_speed}$
0_0 blue	0.16517	0.16517	491.29	923.64	421.22	1032.5	40.973
0_1 purple	0.16583	0.16583	758.82	1166.3	723.42	1215.5	41.298
1_0 yellow	0.16467	0.16467	496.34	909.46	434.83	1008	49.4
1_1 red	0.16567	0.16567	769.76	1157.8	706.16	1180.6	49.7
	$\overline{max\_accel}$	$\overline{max\_decel}$	$\overline{cur\_speed}$	$\overline{sum\_stand}$	$\overline{mean\_stand}$		
0_0 blue	1.7464	4.4975	7.3889	39012	0.98553		
0_1 purple	1.7704	4.5215	7.0142	87467	0.99019		
1_0 yellow	2.964	5.928	8.8247	30828	0.98202		
1_1 red	2.982	5.964	7.491	83743	0.9886		

XY Position and XY Destination

The linear regression model for the position parameter and the destination parameter does not provide additional benefit for the analysis.

For the sake of completeness, in the Appendix C.4.6, Figures C.9 and C.10 pertaining to position and Figures C.11 and C.12 pertaining to destination describe the coordinate system, thus, only roughly the angled map in Figure 6.7. These xy positions and xy destinations represent the average actual positions and destinations for each coordinate, but not as a vector. Therefore, route segments are describable but only if the network is normed and turned using Hildesheimer Street as the y-axis. An interesting graph is illustrated in Figure C.12: the different scenario and location lines almost grow into each other, because the vehicles all exit at an equivalent level.

Maximum Speed

Linear regression model for parameter - maximum speed:

$$max\_speed \sim 1 + scenario * location \tag{7.9}$$

The maximum speed is measured in km/h in Figure 7.13 with similar progress as the maximum acceleration measured in m/s. The progress verifies the intuition: once accelerating, the speed increases. There are three lines: the red line stands for the Hanover Südstadt heterogeneous scenario and the blue line, visible at time step 900, for the Hildesheimer homogeneous scenario blue line Interesting is the constant horizontal purple line representing the Hanover Südstadt homogeneous scenario with the speed of $50km/h$, which is the speed limit in urban areas in Germany. This indicates that those vehicles can drive through the network without stops at an optimal speed. The yellow line has a maximum velocity of $70km/h$ and, therefore, is not within the scale, which ends at $65km/h$. In Table 7.7, the average speeds in the different locations Hanover Südstadt and Hildesheimer Street are slightly higher with $49.7km/h$ for the red line and $49.4km/h$ for the yellow line.

Maximum Acceleration

Linear regression model for parameter - maximum acceleration:

$$max\_accel \sim 1 + scenario * location \tag{7.10}$$

The maximum acceleration in Figure 7.11 is difficult to decipher, because there are three lines: the red line stands for the Hanover Südstadt heterogeneous scenario and the blue line, visible at time step 900, stands for the Hildesheimer homogeneous scenario. Additionally, there is a constant horizontal purple line representing the Hanover Südstadt homogeneous scenario. The yellow line is below the red line with an acceleration of 0.8 (according to vehicle definitions explained for homogeneous vehicles in Section 6.3) but not within the scale, which starts at 1.5.

Maximum Deceleration

Linear regression model for parameter - maximum deceleration:

$$max\_decel \sim 1 + scenario * location \tag{7.11}$$

The maximum deceleration in Figure 7.12 is difficult to decipher, because there are only two lines: the red line stands for the Hanover Südstadt heterogeneous scenario and the blue line, visible at time step 900, stands for the Hildesheimer homogeneous scenario. From Table 7.7, the numbers show that the homogeneous Hanover Südstadt 5.964 decelerate more often than their heterogeneous equivalent 4.4975, which also relates to the current speed. This means that, in the homogeneous scenario, groups are more advantageous for the vehicles' journey. But, for example, in electronic vehicles where there is energy recovery from braking, then the heterogeneous vehicles might regain more energy depending on the braking power.

The second frequent line is the heterogeneous scenario at the Hildesheimer Street location for decelerating 5.928 from Table 7.7,.

Current Speed

Linear regression model for parameter - Current Speed:

$$cur\_speed \sim 1 + scenario * location \tag{7.12}$$

The current speed in Figure 7.14 is a significant plot depending on the location. The heterogeneous scenario at the Hildesheimer location has the highest amplitude of speeds, 8.8247 from Table 7.7, and the highest total velocity in general. The purple progression has the lowest speed with 7.0142, representing the location Hanover Südstadt, homogeneous scenario. The blue line has more amplitude with 7.3889 than the red progression, which, at 7.491, is in the middle range.

However, the homogeneous scenario falls a little behind; there is a gap between the heterogeneous scenario and the homogeneous scenario with less speed. In contrast, the homogeneous scenarios in Hanover Südstadt are faster compared to the heterogeneous scenarios, but in total with less difference between their amplitudes, in fact, almost identical. This points to the hypothesis that faster vehicles can distribute themselves better in the urban network than similar vehicles which stay together more compactly in their routes as well as their speeds. The heterogeneous vehicles drive the most distances, which is logic: the faster the driving, the more distance can be covered in the same time.

Total Stop Times

Linear regression model for parameter - Total Stop Times (sum of standing times)

$$sum\_stand \sim 1 + scenario * location \tag{7.13}$$

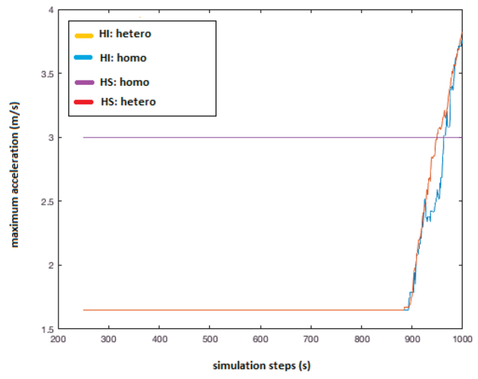


Figure 7.11: Maximum Acceleration.

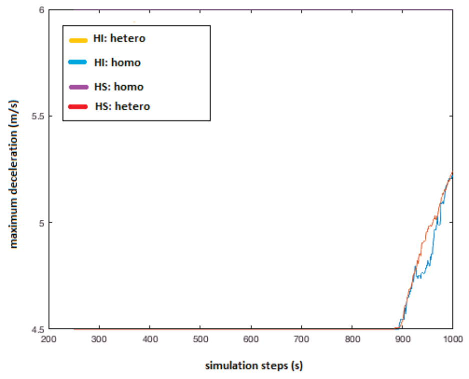


Figure 7.12: Maximum Deceleration.

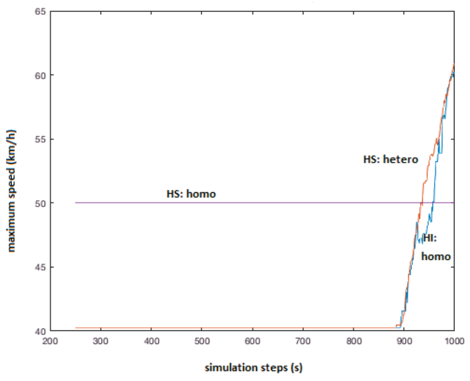


Figure 7.13: Maximum Speed.

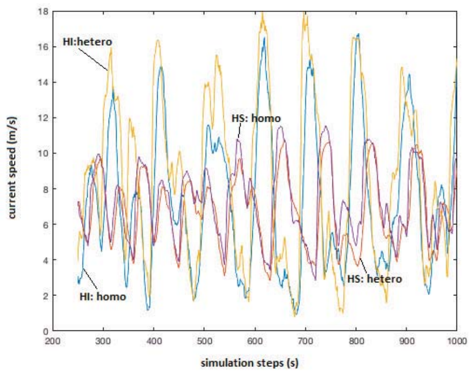


Figure 7.14: Current Speed.

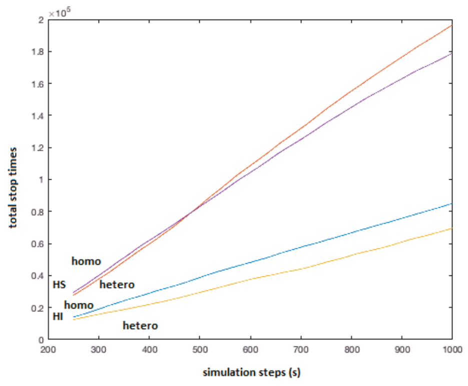


Figure 7.15: Total Stop Times.

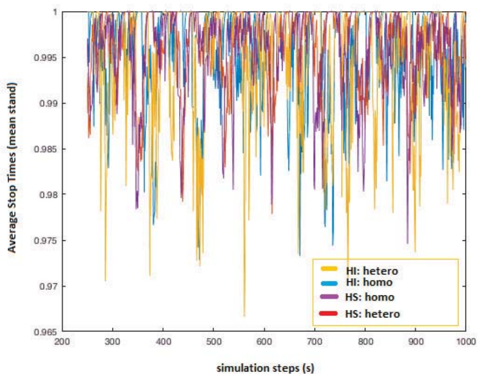


Figure 7.16: Average Stop Times.

The total stop times display more stops as a linear progression, which is reasonable: the longer the route, the more stops there are, and the bigger the location, the more stops it has. Interestingly, the Hanover Südstadt location heterogeneous scenario has, in the beginning, less stops, but outdistances the homogeneous scenario at time step 500. In contrast, at the Hildesheimer location the homogeneous scenario has more stops in general than the heterogeneous scenario. This indicates that more variation for vehicle types is better for the network, which relates to hypothesis 7.1.2.

Average Stop Times

Linear regression model for parameter - Average Stop Times (mean standing times)

$$mean\_stand \sim 1 + scenario * location \tag{7.14}$$

The phenomenon of the summarized stop times is shown in Figure 7.15 with mainly 1 stop for all scenarios varying over time steps with peaks where it is possible to drive through the network without stopping. The yellow line stands for the heterogeneous Hildesheimer Street combination. The yellow progression has, on average, the biggest peaks in Figure 7.16, indicating the minimum of 0.98202, and the purple line has the maximum progression with 0.99019 from Table 7.7. Note that the range of the mean standing times is very small. The Hildesheimer location with both homogeneous and heterogeneous scenarios performs better on average due to the green wave optimized location.

7.5 Discussion of Outcomes

The idea of this thesis was to use decentralized autonomous vehicles for group formation in urban traffic in order to contribute to global and individual efficiency and observe its effects, such as distributing the vehicles in the network.

Agents were used as the modeling paradigm to reflect environmental changes and decision making including communication and the ability to use organizational concepts. The tool supports the modeler and gives many degrees of freedom. The search for a tool for simulating vehicle groups in traffic simulators lead to a proprietary development called MATI. A formalized approach alongside the group life cycle was used. The implementation developed and use cases were implemented. The solution of autonomous vehicle group formation was designed in order to compare similarities of vehicles which then join a group if similar and drive with smaller gaps between them.

Experimental simulation contribute to answering the four field research questions and prove the hypotheses for evaluation, namely, group efficiency, group similarities, group formation and group parameters. Usually, simulations can be benchmarked. Since AVGF is a proof-of-concept with its own framework and simulation tool it is a bit difficult to compare. A similar proposal are platoons on highways, which is featured here.

Benefits and drawbacks are discussed in this section, related to the results of the research questions and hypotheses 7.5.1. The general outcome 7.5.2 is discussed as related to the evaluation criteria 7.2.2.

The general distribution in the statistical graph plots is the same for the distances 7.4.2 and group formation 7.4.3. For instance, the red line, indicating the heterogeneous scenario at the Hanover Südstadt location, always has five major peaks, whereas the purple progression at the same location for the homogeneous scenario has a level progression until time step 650 and then the trend reaches two lower levels. From the Hildesheimer Street location, the yellow heterogeneous scenario overtakes the blue homogeneous scenario slightly every now and then after 700 time steps.

The trend is identifiable that, at the Hanover Südstadt location, the distances of the homogeneous and heterogeneous scenarios start close together and then diverging after time step 650, whereas, at the Hildesheimer location, the trend is reversed, with more diverse values in the earlier time steps which converging after time step 700.

The simulations validate vehicle groups as an allocation technique for urban traffic networks with many advantages over centralized control, such as the complete distribution and the computational tractability. However, in general it is not always possible to have perfect knowledge of the traffic demand or the weights of the street segments (cost functions). Therefore, in the experiments, realistic data and assumptions were used and adapted for the artificial scenarios: the traffic demand was set at a LoS C-D and a digital map of the environment was provided to every vehicle as a priori knowledge.

One problem which occurred was the data logging using calculation time, which can be optimized. Thus, the choice of simulation data logged was too limited and, in the beginning, too little information was logged. The grouping parameter was added for identifying the groups after the simulation and coloring them for the evaluation. While analyzing the data records, big matrices with lots of zero values were produced through the simulation. This could be optimized in such a way that the same results along the diagonal could be reduced to half. Taking out zero values could also optimize the matrices to less than half the size.

Speed of solution can be optimized. Standard hardware was used. The virtual machines for the experiments threw a library error, which needed to be solved to run the scenarios.

Notice that the empirical data is consistent with the theoretical predictions modulo the fact that the autonomous vehicle does not have the ability to reach every green light at full speed. The deviations between the lines fulfill the expectations. The interpretation is time vehicle drive until stopping at an intersection and then accelerating again to full speed. Thus, the simulator and the autonomous vehicles, represented by agents, operate correctly.

The values of the statistical analysis are absolute values and, compared to the realistic dimensions, they might be overweight.

7.5.1 Discussion of Research Questions and Hypotheses

For the experiments, several use cases were created and used as different location scenarios. The noteworthy results are summarized in the following. After the results of the four blocks, related research questions and hypothesis 7.1 are presented.

3-Lane Scenario The lower boundary in the experiments, the 3-lane scenario is as ideal, because no slowing down at the intersection is required. Additionally vehicles traveling the same direction usually do not block each other since all vehicles drive at the speed limit. Just in heterogeneous scenarios, different vehicle properties have an effect of different speeds. For this reason, every vehicle in the 3-lane scenario experiences 0 delay. Many different runs were executed with the 3-lane scenario to verify that the MATI simulator worked as expected. By varying the characteristics of vehicles,different delays can be produced with the 3-lane scenario. Applying the scenario to real-world vehicles usually drive faster than the speed limit indicates.

Green Wave Scenario: Hildesheimer Street To evaluate the delay with the HI green wave scenario, several different signal phases were set and ran for 1000 simulation steps for an increasing vehicle spawning probability from 0.001 to 0.02. The results show that for lighter traffic, longer phases of traffic lights are better. In many cities this insight is already known and is used daily. For late at night the traffic light phases are adopted accordingly. As the traffic increases, higher frequency of phase switches are more efficient (cf. [89]). Eventually delay begins to increase very rapidly and the traffic is hard to handle. The delay levels off because the autonomous vehicles are so backed up that the simulation cannot create any more. An interesting thing to note is that, even with intersections coordinated for a green wave, the vehicle group-oriented system does not break down until reaching a much higher level of traffic. Overloaded group-oriented driving is chaotic: exactly when and how it breaks down varies with when vehicles are spawned and in which directions they are traveling. Running the simulation produces different amounts of total delay for a fixed number of steps.

This verifies the research question 7.1.3 and the second hypothesis, that group-oriented driving is most useful with slightly congested traffic and less so at night or in jammed traffic.

Network Scenario: Southern Part of Hanover To demonstrate the drastic improvement in throughput with the group-oriented algorithm, the same traffic amounts were run as in the HI green wave scenario, but this time group-oriented driving was used with a heterogeneous scenario. This results in improvement over the green wave scenario by increasing distribution to the homogeneous scenario, verifying the hypothesis of group formation 'Homogeneous vehicle groups are more effective in throughput than heterogeneous groups at optimized green wave locations' (see 7.1.2). The amount of traffic handled by this heterogeneous simulation is much higher, and the associated delays are much lower. Note that, with the heterogeneous scenario, it is permissible to have more than one vehicle in the intersection at the same time.

Grid Scenario: Manhattan The question the Manhattan grid scenario should answer is whether or not the MATI simulation scales up to larger networks. The tests go up to 100 intersections (10 intersections in each dimension), and the simulation broke down with more complexity, so the performance level used too many resources. Also, the traffic density did not scale very well. Note that the MATI system was not designed for large simulations with many vehicles and traffic lights.

Group Effectiveness

Groups are not either effective or ineffective, but group effectiveness is studied on a continuum. The model for group effectiveness is measured through three factors: group context, group structure, and group process. Each factor comprises its own elements as pieces of a puzzle that need to fit together for the group to be effective, i.e., technology and material resources, motivating task and conflict management have to be effective and the relationship between the elements must be congruent. Three criteria contribute to making a group effective: individual abilities, group process and performance.

- **individual:** group members expect that through the group work, they also fulfill some of their personal needs.
- **process:** refers to how the group is formed rather than what is done. Processes and structures need to enable joint work and enhance the ability to do so in future.
- **performance:** uses the expectations and satisfaction of the group's internal and external assessment to determine whether its work is acceptable. For effective group work, all above-mentioned criteria need to be met, because they are interrelated.

The metrics of throughput and delay, presented in Section 7.4.1, affect the total travel time cost paid by the individual and, ultimately, by the whole population of vehicles. Concerning the research question 7.1.1 RQ1 (Can autonomous vehicle groups contribute to traffic efficiency?), the answer is that autonomous vehicle group formation reduces the mean travel time by up to 14% and the number of stops per vehicle by up to 20%. This improves global and individual use in urban traffic. If vehicle groups are formed, then the throughput will be improved by 10 to 20% for the individual and for global use compared to the default behavior, depending on the network. Therefore, it is better for the individual to drive in a group. Further details of the economic benefit are described in the next paragraph 7.5.1.

The process of group formation is reflected in the results of efficiency 7.4.1, but mainly in the group formation section 7.4.3. The group process requires vehicular communication. The network and the number of vehicles influence the number of groups and group size. The algorithm works with all vehicles in the network: with, on average, 200 vehicles, there are 25 groups of about 8 vehicles each at the HS location and, with an average of 100 vehicles, there are about 12 groups at the HI location. The realistic group size in urban traffic was determined to be seven to eight autonomous vehicles, which is fairly optimal compared to the lower limit of highway platoons (a platoon consists of 8 to 25 vehicles). In the homogeneous scenario, the group size can be less with a lower limit of 6 vehicles. More details are described in the following Section 7.5.1.

It is possible to appreciate how the grouping performance minimizes the social costs of transportation, very closely approaching the minimal cost to society. Therefore, the grouping strategy benefits the global and individual goals of minimizing travel and stop times by using the street capacities of the available network almost ideally.

The simulations for autonomous vehicle groups were performed with different dimensions. The results of the speeds and delays showed that the autonomous groups outperform the individual default behavior in all the tests, with some more benefits. The autonomous groups allow the vehicles to drive faster and, therefore, to arrive at their destination with an improved travel time, which was the goal of this thesis. This proves the hypothesis 7.1.1 that, if vehicle groups are formed, then the travel time / throughput will be improved by 10-20% for individual and global use compared to the default behavior, depending also on the network. Also, the stop times /delays were reduced by forming autonomous groups. A further positive effect of the autonomous vehicle groups is that delivery times could be more reliably predicted.

To summarize, group formation and coordinated operation of autonomous vehicle groups have a positive impact on travel efficiency in urban scenarios as measured in the locations HI and HS. This strongly depends on the proportion of vehicles that are equipped with the capability of V2X communication and group formation, therefore the decision to use only autonomous vehicles for the simulation experiments gives the best results. The evaluation does not use different penetration rates of different stages of autonomous vehicles (i.e., from mechanical vehicles with driver overhalf-automated to fully autonomous). This was not the scope of this thesis, because the full effect of grouping can only be verified if only completely autonomous vehicles are used.

The metrics of throughput, delay and times all work toward approaching the maximum flow in the urban traffic network N. The maximum flow problem (cf. [269] p. 365) is solvable in polynomial-time seeming inherently sequential. Therefore, it has a crucial weakness for parallel computation, because it works in stages. At each stage, a flow f is started, which has the everywhere-zero-flow in the first stage, and is then improved on. To this end, a new network $N(f)$ is constructed, reflecting the improvement potential of the streets of N with respect to f, and trying to find a path from the source s to the sink t (the origin and destination) in $N(f)$. If it succeeds, then the flow is improved. If it fails, the current flow is the maximum. It is not hard to see that each stage can be parallelized most satisfactorily. With enough hardware, $N(f)$ is constructed in a single parallel step. Thus, each stage can be done in $\mathcal{O}(log^2 n)$ parallel time and

$\mathcal{O}(n^2)$ total work, where n is the number of nodes (intersections) in the network. The problem is that stages need to be carried out one after the other, and the number of stages may be very large - certainly more than poly-logarithmic in n.

Although the group algorithm for throughput and delay presented in Section 7.4.1 is more like a role-based contract net protocol as described in `Fiosins et al.` ([110] p. 78) compared to the dynamic AEDF algorithm, it represents autonomous vehicle group formation in the locations HI and HS with similar results on travel and stop times due to the same measured traffic density.

Other experiments (reported in `Görmer et al.` [137]) indicate that the benefit of autonomous grouping depends on the overall traffic density, the so-called Level of Service (LoS). Best results can be achieved for dense but not congested traffic. Whereas in light traffic, too few vehicles are available to form groups, in a traffic jam, the vehicles are already involuntarily grouped, so in these situations grouping does not improve the traffic situation. Therefore, the experimental settings were scenarios with dense traffic measured in Hanover, Germany.

Economic benefit The economic benefit is bigger with group strategies for efficiency, comfort, and safety as stated by the benchmark of vehicle platooning on AHS. The travel time was analyzed before in Section 7.4.1. Other indicators for measuring the economic benefit with autonomous vehicles are fuel consumption and safety.

Autonomous vehicle groups smooth the traffic flow by minimizing the human-made stop-and-go effect that leads to higher emissions and congestion. This effect is amplified when vehicle groups are allowed to re-route and dynamically form groups. AVGF leads to higher speeds on the streets, due to coordination and faster reaction times. This research focused on actual implementations of the proposed group strategies rather than on the effect of potential. The overall traffic flow is smoother and faster, resulting in lower emissions.

Individual urban traffic flow-density-speed relationships are mathematically approximated by the safe gap between two vehicles so the second one can react without collision to the first one stopping. In Figure 7.17, the lower flow-density curve indicates the first car stopping at 0.9 g (8.82 m/s^2) and the second car braking at 0.6 g (5.88 m/s^2) after one time lag. The distribution is for a single lane and is similar to the predictions of traffic manuals. A similar mathematical result is produced by the vehicle groups with the size of eight vehicles in urban traffic where the gap is much smaller due to autonomous group regulations. Thus, grouped vehicles travel faster and more compactly, which improves the traffic throughput. The two curves diverge significantly from each other and reflect the traffic density and flow associated with specified safe gaps at the vehicle speeds measured. The maximum traffic flow for autonomous vehicle groups is approximately four times that of the individual default behavior. The

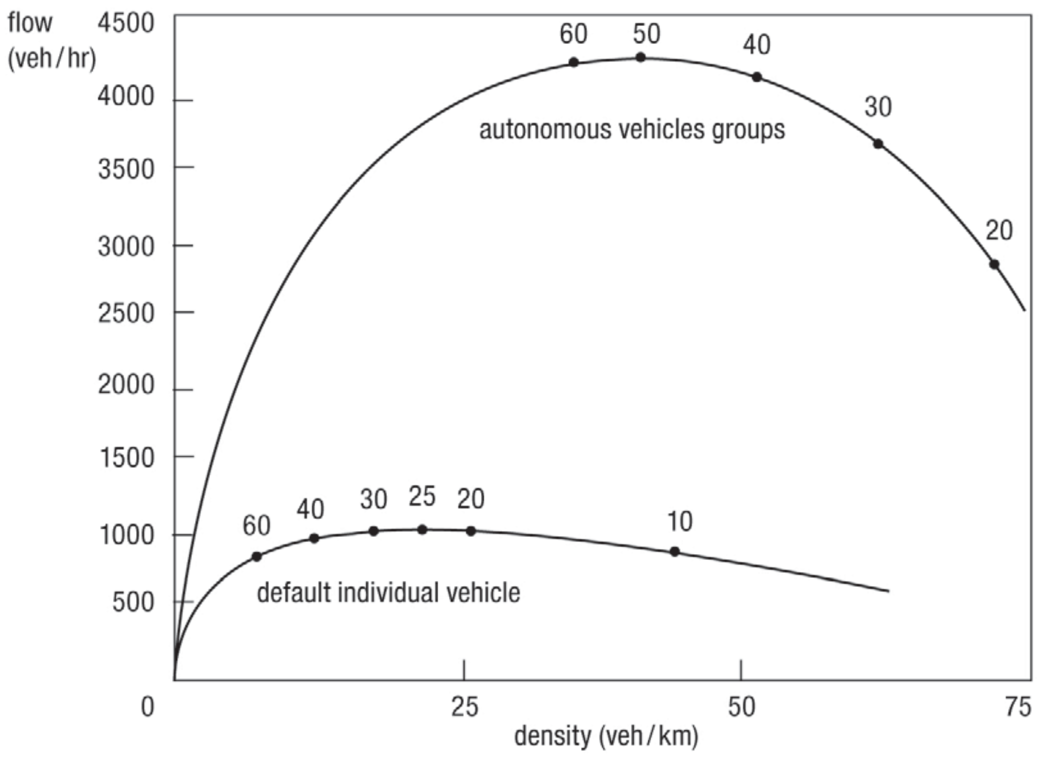


Figure 7.17: Flow-density-speed relationships.

maximum flow for autonomous vehicle groups appears at an average speed of 50 km/h whereas for the individual case it occurs at 25 km/h.

Due to the smaller gap between group members, the aerodynamic traction is also reduced for the following vehicles. Therefore, the power demand on the engine and consequently the emissions are smaller when grouped.

Linking total CO emissions to traffic flow on a one-kilometer stretch of road is illustrated in Figure 7.18 for the individual default and for autonomous groups. The maximum individual traffic flow is 1100 vehicles/hour. At the same traffic volume with group sizes from three to eight vehicles, the CO emissions are approximately half. Almost twice the traffic volume occurs with groups for the same amount of emissions compared to the emission rate for maximum individual traffic. Thus, the emissions connected to the maximum traffic flow for vehicle groups are approximately half of the individual ones. In these curves, the emissions associated with higher traffic densities and lower average speeds are underestimated because the calculation is based on steady-state speeds. The accelerations correlate to stop-and-go traffic which leads to more emissions.

According to the Platooning Program for Public Motorways (SARTRE) [294], a benefit of 20% is estimated in fuel consumption which is adopted for groups, varying depending on the number of vehicles in a group, the gaps between vehicles and the aerodynamics.

SARTRE states that drivers' inattention causes 18% of road fatalities. Au-

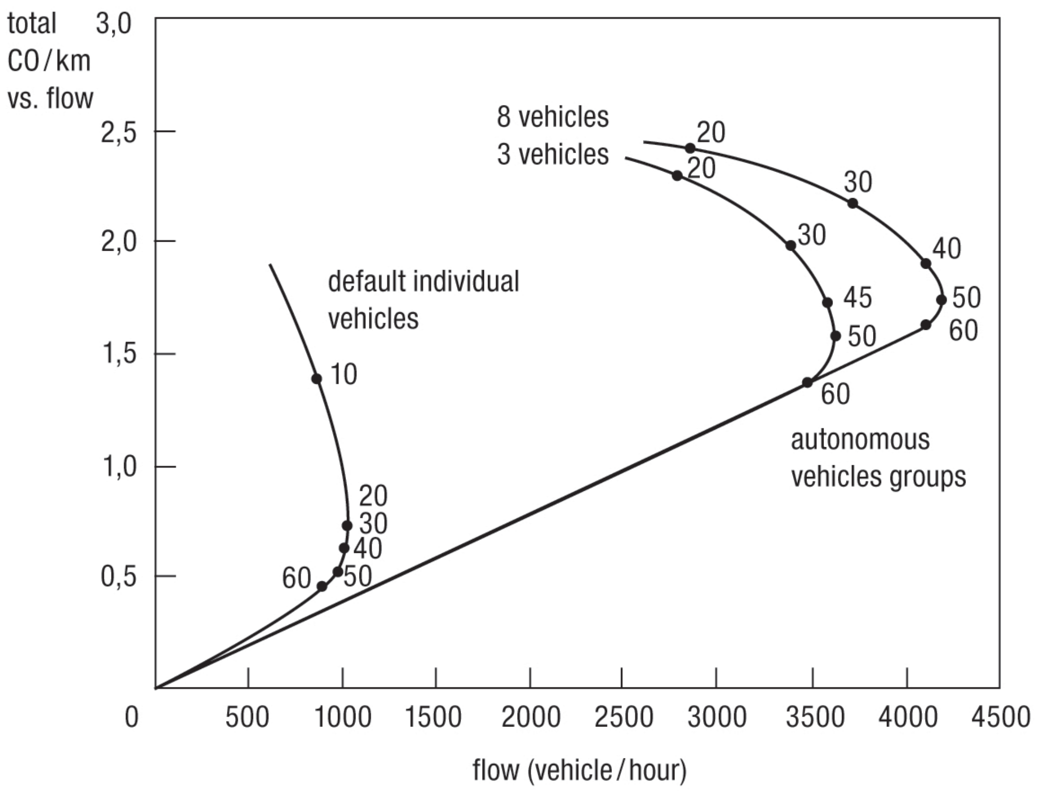


Figure 7.18: CO Emissions.

tonomous vehicles without human failure improve safety by an anticipated 10% reduction in fatalities (without the group effect) and also result in increased driving convenience.

Thus, AVGF results in environmental, safety, congestion, and convenience benefits like the benchmark proposed.

Group Similarities

The dynamic vehicle group strategy is especially useful for individuals because their individual utility can be maximized and is not a source of internal conflict. Additionally, AVGF is more practical due to less traffic planning in group phases, as a static or pre-planned algorithm would require. A static definition of the first phase of group formation takes longer compared to the main phase and the post phase.

In dynamic group formation, the transition is fluent due to the previous settings of the algorithm, but needs more calculation. AVGF avoids coordination booths for grouping through its algorithmic implementation. Thus, no real coordination or communication times hinder the traffic and no extra infrastructure needs to be built.

Group formation is defined and measured in three phases. With AVGF, it is not possible to measure how long the phases take because they are done in a continuum. This is why the simulation takes some time, as stated in Section 7.3.2, but also all variable attributes are calculated, which slows down the simulation.

The hypothesis 7.1.2 can be verified: If the traffic scenarios are more complex and dynamic, the more dynamic group strategies improve the individual and global use. The effects on HI compared to HS were presented in Section 7.4, especially in the Subsection 7.4.2. Additionally, the effects of homogeneous and heterogeneous groups at the locations HI and HS were plotted. Homogeneous vehicle groups always outperformed the heterogeneous groups in the optimized Hildesheimer green wave (HI) scenario, as opposed to the location Hanover Südstadt (HS), where, after a warm-up of 8 minutes in simulation, the heterogeneous groups outperformed the homogeneous groups. This verifies also the other hypotheses 7.1.2 regarding group similarities.

Heterogeneous vehicle properties showed a negative impact on the average speeds, which is expected realistic behavior. In order to improve, it is recommended to form groups with very similar properties within the group, so that the differences between the vehicles are as small as possible. The tests showed that, the bigger the differences in properties between the vehicle classes, the better the speed and delay for the autonomous vehicle groups. Therefore, a big variance between groups is recommended.

These results with heterogeneous vehicles are relevant in reality, because cities in Germany have mostly small vehicles (which are easier to park) and a few long vehicles in the form of public transport buses or long delivery trucks. Normal and fast vehicles are not as common. Therefore, also the types of heterogeneous vehicles have a tendency to be quite homogeneous and only divege by 8%.

The composition of the heterogeneous vehicles was assumed using the trend of vehicles in Germany, but not accurately fitting to the location of Hanover or to city traffic. A spot test in rush-hour traffic in the Southern part of Hanover counting the four categories between 7:30 and 8:30 a.m. in in-going traffic would make the distribution of vehicle types more realistic for the location. Any other appropriate reasonable selection with the derivation for the impact on reality could give different results. But, for a first tendency, the heterogeneous scenarios gave a reasonable result to the research questions and hypotheses.

For future use, variations of heterogeneous vehicle types could be investigated in more detail.

Simulation Default versus Homogeneous versus Heterogeneous One question that arises is whether or not the vehicle characteristics of speed and length have an impact on the group-oriented driving. Therefore, in addition to the standard normal vehicle for homogeneous vehicles, a small and long vehicle, as well as a fast one were included in the heterogeneous experiments.

The Figure 7.14 shows the speeds of all autonomous vehicle classes. The phenomenon of the distinct speeds of the different autonomous vehicle classes shows clearly, as expected, due to their different speed settings. The average speed of the heterogeneous vehicles is always higher than that of the homogeneous vehicles, illustrated by the yellow and red lines respectively. Compared to the homogeneous vehicles (blue and purple), there is a clear distinction also between the locations HI and HS. The homogeneous HI blue amplitude mostly stays below the equivalent HI heterogeneous yellow curve, which has the highest speed differences. In contrast, the HS homogeneous simulation, represented by the purple curve, has a bigger amplitude, being faster and slower than the HS heterogeneous red line, which is more steady in speed.

Additionally, evidence from Figure 4.12 suggests that the autonomous vehicles drive faster than the average of the individual default vehicles of the simulation. The differences between autonomous vehicle groups and default simulation vehicles have a big influence on the delay/ stop times. The phenomenon here is that the delays of all vehicle classes are very close together in the range of 0 to $5s/km$. The average delay of the autonomous vehicles of the normal class (green horizontal line) is about $1.8s/km$. It is remarkable that the delay times of long vehicles are close together in the range of 0.5 to $2.2s/km$. This is significant because it means the delay time of a random long vehicle is, at most, $1.5s/km$, which is good for logistics to be able to calculate exact delivery plans.

On the other hand, the average delay of the individual default vehicles shows that the delay times of the long vehicle class are distributed between 0 to $15s/km$. The average delay of the individual default vehicles of the normal class is about $5s/km$ (green horizontal line). Considering that the random delay time of a long vehicle is $10s/km$ at most, the logistics delivery plans have a high variation and are imprecise as a consequence.

The small vehicles have a special significance, with a delay of $0.4s/km$ for autonomous vehicles, but the individual default vehicles do not have a delay. The reason is that the autonomous vehicles lower their speeds in order to change to group lanes - thus, as a consequence, more dynamic grouping using all available lanes is preferred compared to group activities like platooning [294] using group lanes. On average, the delay times of autonomous vehicles are $7s/km$ lower than the average delay of individual default vehicles.

The results of speeds and delays have shown that autonomous vehicles drive faster to their destination. The experiments were configured in such a way that there were no differences in vehicle properties, which is artificial, but helpful for first indications. In the real world, there are always differences between vehicle properties.

That is why homogeneous vehicle groups are better suited for realistic scenarios and heterogeneous vehicles can be configured. Thus, vehicle properties of the same class are more similar than the vehicle properties of different vehicle classes. For example, vehicles of one group could have a maximum difference of $10km/h$ and the acceleration and deceleration could vary by $0.5m/s$. The average speed of autonomous and default simulation vehicles of $49.6km/h$ is only slightly different compared to heterogeneous groups at $50.5km/h$.

This is explainable with the configuration of simulations of traffic dynamics which should be as realistic as possible. Another reason could be that the vehicle classes block each other due to the different vehicle characteristics. As a consequence, an order of groups with the fast in front and the slow behind would be preferable, but is not very realistic and realizable.

Compared to the individual default vehicles, the autonomous vehicles arede-layed less . The average delay of the fast class is $7 sec/km$ for autonomous and $12.5 sec/km$ for individual vehicles. That means an autonomous group vehicle exceeds the individual vehicles by approximately $5.6 sec/km$ and is faster.

In general, it can be summarized that an autonomous group gets delayed by $3.6 sec/km$ whereas the default vehicles have a delay of around $5.7 sec/km$.

Heterogeneous vehicle groups performed well in all the scenarios, averaging less than one half of one percent for the delay in the green wave scenario. These results are presented in Section 7.4. Recall that the 3-lane scenario is represents the lower limit which leads to 0 delay.

This heterogeneous test shows that small differences of properties still have an effect on autonomous groups in speed and delays compared to default simulation behavior.

Group Formation

The network influences depend on the amount of traffic lights, free-driving segments for coordination, number of lanes, and speed allowance.

In the following Figure 7.19, the vehicles are plotted according to their time step on the abscissa and their grouping parameter on the ordinate. Not all vehicles are in the plot at the steps, but coming in later time steps. Some vehicles stay in their groups as desired and others, like the orange vehicle 24, change back and forth into three different groups. Zooming in will explain the reason for the sometimes frequent changes into other groups. There are several parameters which result in the grouping parameter, but since this is a spot check on homogeneous groups, only the actual position with its x and y coordinates affects the group change. In the beginning, vehicle 24 is in the same group as vehicle 12, but changes after time step 25 into the next group, together with vehicle 7 in time step 37. After time step 75 to 83, vehicle 24 changes into a close group where its distance to the group before is minimal. This shows that the vehicles group themselves dynamically. Additionally, the group size in the scenario of Hanover Südstadt is limited to a maximum of 8 vehicles, which is the result from pivot tables in the different time steps. The more time steps exist, the more groups there are, which have a varying group size between three and eight. This is due to environment restrictions of street lengths limited by cutting points of intersections and signal plans limiting the green phases, since the experiments are done with fixed signal plans. Therefore, an explanation of the last group change could be that the group before was full and a similar group with similar parameters was initiated.

Regarding the research question7.1.3: Which impact do homogeneous and heterogeneous vehicle properties have on the vehicle groups? Are there differences in groups?

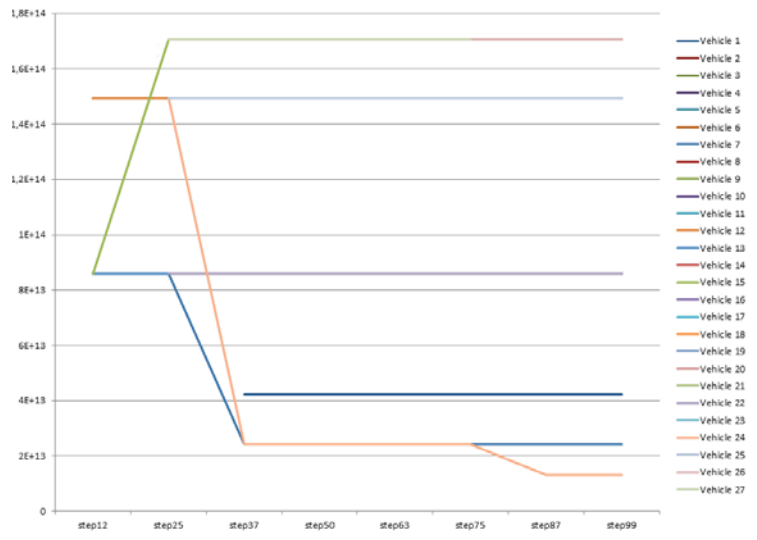


Figure 7.19: Results of Vehicle Groups in Scenario Hanover Südstadt.

The more homogeneous the vehicle properties, the bigger the vehicle groups. Within a group, the properties should be as homogeneous as possible, whereas between groups the properties should be heterogeneous to distinguish the groups from each other and also to maintain group stability for a longer period.

Different criteria of group formation in the four scenarios are mostly the technical aspects. In general, the scenario settings are modeled identically and adjusted to the different complexities of scenarios like number of agents, simulation steps and different initial random runs. In realistic scenarios, the difference in group scenarios is small, whereas the highly scalable Grid has performance issues. That is why only realistic scenario locations were evaluated.

Surprisingly, autonomous vehicle groups contain up to 8 vehicles in a group, which is quite ideal for urban traffic and even reaches the lower limit of the AHS group size of eight vehicles.

Group Parameters

The group parameters used for the AVGF algorithm are the maximum speed, acceleration, deceleration, actual position, desired destination, and individual route. Those parameters come from simulation and are clustered into one similarity value to match the individual vehicles to similar vehicle groups.

The hypothesis 7.1.4: "The more variable parameters there are which can effect the traffic, the more ways the results can be interpreted" is in general correct, but can be falsified for the scenario and location combination tested. It is possible to extract clear tendencies and trends.

In order to evaluate the vehicle groups with the default individual driving, the dissimilarity between simulated speed and desired speed is calculated and logged for the vehicles. The driving method with the smaller variations in similarities is therefore beneficial for the vehicles, either grouped or individual. Thus, AVGF is beneficial for urban traffic.

Whereas pre-planned group phases minimize variables, different aspects and details still need to be isolated for measuring. This is why group phases are not relevant once again. Especially with AVGF, the phases are fluent and the length of each phase cannot be measured due to individual joining and leaving. The vehicle joins a group if the similarity value is in the range of accepting, and as a group action smaller gaps between vehicles are allowed, but each vehicle can leave the group at any time depending on its utility.

The results of the static group sorting are very predictable, but therefore not very realistic. This verifies the hypothesis 'If the phases are more planned, then it adapts less to real-word traffic and is more artificial.' Technical parameters can be tested on simple and predictable scenarios, but dynamic, complex and realistic behavior is eliminated through the very planned model specifications.

If the vehicle properties are more homogeneous, then the vehicles join groups. The independent parameter was the actual position mainly, but other major influencing parameters were current speed and amount of stops. Surprisingly, the destination was minor, as well as maximum speed, acceleration and deceleration of a vehicle. Dependent parameters were the infrastructure location and scenario configuration.

Alpha-Heuristic Discussion In Section 6.5, different value functions are discussed including the alpha value, especially in Equation 6.2. This condition sets the meaning of the value, so that the total distribution stays the same, but individual weights of the functions can have more effect than others, so-called classification factors.

Within homogeneous or heterogeneous vehicle characteristics, no difference occurs between the vehicles of the same class. Thus, the change of the alpha value affects groups with heterogeneous characteristics where a configuration of the alpha values makes a difference besides the class characteristics. The property differences of the same class are significantly smaller than the properties of vehicles of different vehicle types. The heterogeneous vehicles types were configured in Subsection 6.3.2. The configuration of the alpha value makes it possible to adjust whether two vehicles of different groups can group together if desired. Otherwise, the standard configuration just lets vehicles of the same class type join the same group.

Desired speed can be set as the priority so all possible members could join, whereas the parameters of acceleration and deceleration are only relevant for groups of the same class. For example, $\alpha = 2$ for speed is set to double, whereas the acceleration has a reduced $\alpha = 0.4$, which is less relevant than the ability to stop for deceleration with $\alpha = 0.6$. The three alpha values need to be, in total, the combination of all weighing factors, in the example three values are adjusted and the total is 3.

The dynamic grouping in the AEDF has seven alpha values:$\alpha 1$ desired speed, $\alpha 2$ acceleration, $\alpha 3$ deceleration, $\alpha 4$ the route, $\alpha 5$ the route cost, $\alpha 6$ the actual position and $\alpha7$ the destination as shown in Equation 5.30. The first three alphas have been discussed in the small example before, the other alpha values are put in relation to each other.

The actual position is different for all vehicles of different classes, therefore the tolerance of the alpha value should not be set too small but rather higher if groups with other vehicle types are desired. The route should not have too much impact, because, while in groups, they might change the route altogether or adapt to the route of the leading vehicle. The route costs are important to decide on which route to take, and should have more effect. The change in alpha value does not include the minimization of route costs, just states whether the route costs need to be the same/similar for grouping. The destination is a fixed parameter and groups automatically would form with the same destination, so that can be adjusted to a smaller alpha value for the benefit of other alpha weights on the grouping parameters.

With the discussion on the alpha heuristic, different alpha values can be set depending on the priority of the desired groups. From experiments, the effects of the different values are shown with their similarity factor in detail. As a consequence, it is important to configure vehicle groups so that the differences between the member vehicles are as small as possible while keeping in mind that the goal of groups is to improve the mean travel time and reach the destination as fast as possible, perhaps also by minimizing gaps between group members (which cannot be adjusted by alpha values).

Setting variations on the alpha value can be configured and then, heuristically, the optimal solution for vehicle groups in the scenario of interest can be found.

AVGF shows that, if the vehicle similarities (desired speed, acceleration, deceleration, etc.) between the vehicle groups are big, then the results of travel time and delay are better with the group oriented driving compared to default individual behavior.

This alpha heuristic discussion verifies the hypothesis: 'If the vehicle properties are more homogeneous, then the vehicles join groups.' This is definitely true for vehicle groups and their dissimilarities in different properties. The more similar they are, the more stably groups drive together, whereas the more variances in similarities the groups have, the more flexibly they can join new groups while driving in the network. Studies in the SARTRE project [294] show that up to 3 changes within 20 minutes are acceptable for human drivers, which could be used as a rule of thumb. In Figure 7.19, this assumption is verified.

'When there are more variable parameters which can effect the traffic, then the results can be interpreted in more than one way.' In general this hypothesis is true, but for the group effect the values are compared to their similarities and the results are interpreted towards the group effect in urban traffic. So the interpretation is quite clear.

7.5.2 General Outcome and Further Results

The key criteria are initially assessed on the basis of a six-point rating scale. Scores of 1 to 3 indicate a 'successful' execution, while 4 to 6 are 'unsuccessful'. For the criterion of sustainability only a two-point scale is used. This mainly reflects the anticipated future trend (albeit with a certain degree of uncertainty), where a score of 2 indicates 'insufficient sustainability'.

Relevance

During peak hours in urban traffic where traffic is not distributed wisely there exists the socially undesired state of congested traffic and limited lane resources, which justifies the introduction of autonomous vehicle groups. A group life cycle was modeled in Chapter 5 with three main processes of group planning, formation and operation. Use cases were described in Chapter 6 to fulfill the goal of group effects in urban traffic and were assessed in this Chapter 7. The autonomous vehicle groups were simulated in the MATI traffic simulator and interpreter 4, which was used as a proof-of-concept tool. The design of vehicle groups was fundamentally suited to achieving the goals of this thesis and followed general standards of research. The quality of autonomous vehicle groups follows the assumptions of realistic traffic which means minimizing communication and time for group formation.

There exists a high demand, the theories were respected and the results of vehicle groups have good quality (almost no number of errors) including group cohesiveness, therefore the relevance is rated with the score of 1.

Effectiveness

The surrounding conditions are given to successfully realize the autonomous vehicle groups in the realistic use cases. The implementation was done as described in the model with role-based group leader and members since it is easier to communicate limited information, but also with a dynamic horizontal algorithm based on similarities. Autonomous vehicle groups are fully accepted by the traffic participants and desired because the groups satisfy the individual utilities and the global benefits. Different penetration rates of communication, autonomous vehicles, and group strategies should be investigated in future research.

The effectiveness of autonomous vehicle groups is rated with a score of 1 for the purpose of this thesis, but with a tendency towards 2 because it needs refinements with different penetration rates in order to implement it in future mobility trends. Also, other benchmark algorithms should be applied.

Efficiency

The autonomous vehicle groups are designed and implemented to benefit the individual and global economic use. With AVGF, all vehicles will chose grouping depending on the parameters. The chosen group parameter made the vehicle groups very ideal (full penetration), but different weights could enhance the group efficiency in realistic settings and could be altered for future use.

The cost-benefit-ratio of dynamic group formation is better than individual traffic, thus, the efficiency is rated with the score 2. It was successful, but the efficiency parameter could have more variation.

Overarching Impact

The content and results of AVGF are reusable to solve traffic situations with groups of traffic participants. A situation transfer could also include the traffic lights being grouped in order to give guarantees for vehicle groups. The acquired knowledge about autonomous vehicle groups is transferable to other examples for better use of available resources by autonomous groups of mobile units. Then the requirements can be transferred to robotic situations. Standards like the MATI simulation tool help to make microscopic group simulation possible and could be reused for different environments, interpreters and tools.

The impact is rated with 1, due to the universal use of the simulation, but also AVGF with the similarities based on predefined parameters is relevant.

Sustainability

The aim of this thesis is to show future effects of urban traffic and to provide valuable input for the future of mobility. Sustainability is given by making the research available, but it is a prototype and not a market-ready solution.

Sustainable results are preferred and not only achieve short-term improvements. For this reason, investigations need to be made as to whether any improvements are likely to endure. The criterion of sustainability is considered because, with this thesis, a position is made to successfully continue the promotion of AVGF. Therefore, the contributions of a modular standard simulation tool like MATI and AVGF are sustainable and good for future use.

7.6 Summary

The MATI architecture offers the designer a traffic simulator combined with agents, a wide variety of alternatives for choosing different environments and modeling agents with an interpreter and evaluating them with tools. The main research question dealt with the effects that decentralized and dynamic vehicle groups have on the traffic system as a whole. How useful are autonomous vehicle groups? They are tested as homogeneous and heterogeneous groups for travel time / throughput from three perspectives: the individual perspective, the group perspective, and the perspective of traffic management.

This chapter considered the problem of empirical task allocation for homogeneous and heterogeneous groups, namely, groups of autonomous vehicles. It is well-known that this problem is NP-hard. Therefore, computationally feasible approaches should develop a simple or an approximate solution. Here, an experimental solution was proposed which addresses the latter. This problem was tackled heuristically by combining goal-based group models with role-based hierarchical leader-member relations and dissimilarity functions for clustering the characteristics of each vehicle in relation to others.

The MATI simulator functions partially on the idea that the autonomous vehicle uses whichever traffic management policy is in place. This is similar to real world traffic a red light can be crossed depending on the trigger. However, it is in everyone's best interest to act according to traffic rules, such as signals.

A challenge that remains, however, is modifying the system such that human drivers could use it. Computerized error margins are too small for a human to be able to keep up with the control of autonomous vehicle. In case of enlarging the margins, the efficiency benefits would be reduced, since the ability of an autonomous vehicle agent derive directly from the precise control. In mixed traffic the intersection could be informed about a human driver approaching and responding accordingly to make some extra room for the driver to maneuver through the intersection. For future mobility with autonomous vehicles traffic lights become obsolete, since they can coordinate themselves through communication. Traffic lights are an appropriate control for a high percentage of human drivers.

The state and action space of each vehicle agent in the urban network which operates in the environment simulation was defined. Possible grouping algorithms that are suitable for efficient driving were outlined. No single group method can be the method of choice for all occasions. An optimal program will feature a mixture of instructional methods and cooperative activities.

The experiments are internally and externally valid. Through experimental investigations, a high internal validity is given and the results are clearly interpretable (without alternative descriptions of the results). The experimental design of two artificial and two real-world simulation scenarios offers special conditions of research investigations, so that the results can be generalized for small urban networks, because the model was made to be as realistic as possible in order to be representative. The randomization of vehicles spawned in the networks reduces the disturbing variables. Altogether, this makes the results valid also externally.

The efficient utilization of distributed urban traffic systems remains a difficult problem. Generally, centralized approaches require extensive knowledge of the underlying system and can be computationally intractable and unresponsive to change. Therefore, distributed approaches like groups of robots [86] or computation environments [368] offer an alternative based on artificial intelligence as a paradigm for the design and the control of complex systems. In this chapter, the underlying group model served as a framework for the efficient allocation of urban networks. The collective behavior of vehicle agents was demonstrated, and lead to an efficient use of the urban network depending on the design.

As shown in the evaluation and results, the autonomous group-based approach drastically outperforms the individual driving system. However, many challenges are associated with creating autonomous vehicle groups for the real world. Additionally, within the simulator, there are assumptions in the work reported here that can be relaxed in future work.

What the caterpillar calls the end, the rest of the world calls a butterfly.
Lao Tse

Chapter 8

Conclusions and Future Perspectives

This thesis makes contributions to the comprehensive performance of autonomous vehicle group formation based on an experimental model and a simulative evaluation in urban environments. The focus was on decentralized autonomous vehicle grouping, maintaining the flexibility of single vehicles, but using group coordination for higher throughput in urban networks.

For this thesis, it is assumed that autonomous vehicles (AVs) coordinate themselves better into vehicle groups. AVs make use of calculated algorithms, whereas human drivers deal with physiological and psychological influencing factors. The acceptance of the emerging technology by humans (i.e., less control of their own driving and being in the hands of computer software or the lead driver) and existing regulations and laws may be problematic and make new concepts necessary for autonomous vehicles.

8.1 Main Contributions

The main findings from the performance evaluation study presented in this thesis provided the following major results. This thesis contributes to understanding, modeling, designing or applying innovative solutions for traffic and transportation systems based on agent technologies and approaches. First, a group model was developed and implemented with decentralized strategies. Second, this thesis described and specified agent-based traffic simulators, resulting in MATI, for modeling group formation methods along with metrics for measuring the efficiency. Third, for evaluation purposes, simulations were executed to study the system's dynamics.

8.1.1 Vehicle Groups in Urban Traffic

The model proposes multi-agent group-based vehicles that out-perform the default of individual driving. Subsequently, a technical concept with use cases was designed. This group model was implemented and evaluated in simulation. The conclusion was that the group approach used with AVGF reaches the optimum in simulation since all vehicles are grouped.

This thesis introduces new group-oriented driving methods: one group coordination method with global coordination by the group leader and local decision making of the group members, and the other using the idea of dynamic decentralized group formation based on similarities.

Essential elements in the design and operation of transportation systems are the modeling and simulation methods. The proliferation of interest in Intelligent Transportation Systems (ITS) and Smart Mobility is seen at all levels of government and industry. Transportation researchers have developed models and simulators for planning, design, and operation of such systems.

This thesis investigated whether vehicles can form groups in order to have a positive effect on the traffic throughput. This was tested in a traffic simulation environment with vehicle agents. Simulation results prove that vehicle groups can contribute to a higher throughput in dense traffic. Therefore, this thesis paves the way for distributing traffic control to traffic participants.

Due to the complexity, non-equipped or human-driven vehicles are not considered in the scenarios. Vehicle groups are not treated with priority by traffic management, vehicle groups just follow the recommendations of traffic management. Rules, norms and regulations can also be investigated with the agent paradigm as infrastructure elements of the judiciary, legislative and executive powers. Communication exchange is definitely a catalyst for more decentralization if information is accessible to all participants.

More possibilities can be investigated, because for this thesis communication use is very limited: full communication, no package loss, all vehicles equipped with good performance, wireless communication of the best frequency and shadowing effects were assumed.

8.1.2 Microscopic Agent-Based Simulation Tool

For validating the theory and simulation, state-of-the-art agent-based traffic systems were reviewed. The group coordination was realized based on a newly created platform integrating ATSim (AIMSUN and JADE), whereas the centralized and decentralized groupings were implemented using MATI (SUMO and Jason).

The first simulator (used in the PLANETS research) is based on the commercial traffic simulation software AIMSUN and extends AIMSUNs capabilities by four external applications, i.e., an external traffic control method, a routing module, a multi-agent model and a communication model (V2X). Since the real-time exchange of information plays a critical role in MASs, a communication model was developed, which takes into account the frequency-selective and time-variant character of the radio channel caused by shadowing, reflections and scattering in particular in urban environments. AIMSUN is a commercial traffic simulator needing agent extensions, which needed long development loops. Also, JADE was very generic as an agent interpreter, where everything can be developed, but less standard functions are already implemented.

As a contribution and result, the traffic simulation MATI was developed. MATI couples existing simulation tools to one standardized triple (environment, interpreter, and tools) with its benefits and specialties. For the purpose of this thesis (1) Jason, an exchangeable interpreter which is communicable, with agent logics, capable of coordination, was combined with (2) SUMO, a traffic simulation which is manipulable as the environment and (3) HDF5, analyzing tools for evaluation. Therefore, MATI is a generalizable solution which satisfies all requirements (see Section 2.4) and a tool with the focus to create decentralized vehicle groups in urban traffic, also able to incorporate centralized traffic management input.

Potential applications for this thesis include agent-based simulation of traffic and transportation systems: Applications of agent technology in traffic, transportation, and transport logistics can be simulated and evaluated with MATI, especially with a focus on coordination.

8.1.3 Evaluation of Vehicle Groups

For evaluation purposes, a simulation of the system dynamics was performed, which is able to model the interaction between equipped vehicles and the traffic management system. To support this, as an essential basis for the investigation of various research questions (e.g., concerning the penetration rate of equipped vehicles), the extended AIMSUN simulator was developed.

With two locations and scenarios, dynamic vehicle group formation was demonstrated on the one hand. On the other hand, an evaluation of the quantitative impacts of cooperative decentralized traffic management is given, showing encouraging results. The mean travel time was reduced by up to 14% and the number of stops per vehicle can be reduced by up to 20% with a communication penetration rate of 50%.

Positive aspects of the decentralized approach are the localized effort and speed to address the specific user (autonomous vehicle) needs; within the system it remains bounded in a small scenario and shows an immediate benefit for implementers. Negative aspects are the narrow, vertical viewpoint with the focus on the autonomous vehicles, significant reworking with small system changes (here, modularization and standardization helps) and neglecting cross-boundary influences to a certain degree, which requires expert knowledge.

This thesis encompasses autonomous vehicle group results with analytical discussions, which addresses foresighted new perspectives in the field of dynamic urban traffic, in order to respond to the traffic challenge. Additionally, this work showed some technological potentials of autonomous driving and includes aspects of dynamic traffic management and vehicular communication.

In summary, the *idea* that groups are important in the traffic context was established about 60 years ago. However, only in the past seven years has that idea been taken up to cities and acted on in technology and research in the traffic domain. The emergence of groups in the traffic environment has brought many changes.

8.2 Limitations and Future Work

This last section provides the limitations revealed by the AVGF approach, and opens new research topics with a possible research direction. Lots of challenges need to be addressed, including user acceptance, communication, the multimodality, technology risks and data availability and security to state just a few important ones.

Driving assistance technologies are already intended to support the driver, for instance in moments of absence like microsleep, which can be detected and sound effects used to alert the driver, or cruise control with included distance control, in which the vehicle can decelerate automatically, or line detection along the roads to guide trucks. Smart mobility results in using intelligent vehicles for future use. The phases of getting to a full penetration rate of autonomous vehicles need to be accompanied by models and technologies. Current limitations of this work include the inability of AVs to deal with human-controlled vehicles. Additionally, mixed traffic should be considered, meaning not only vehicles, but other traffic participants or mixed convoys, or micro pods and their interaction. Thus, a research question could be coping with human drivers and mixed traffic for addressing future mobility trends. The idea for depicting this complex field is to join different fields of research to exploit the best concepts. For instance, combining the expertise of politics, the automotive industry and IT knowledge to develop a new mobility standard makes steps towards future mobility transparent, which contributes to a better and safer world.

Additionally a constraint of this work was to characterize AVs into four different stereotypes. Relaxing this limitation and managing the distribution of vehicle types in urban networks is a part of the future research agenda. A research question could deal with the different AV types which are needed for future mobility and a possible distribution of those in urban networks. A possible research direction is to assess the present vehicle types and their distribution in urban traffic on one hand, and, on the other hand, identifying the future mobility trends such as having autonomous car sharing models provided by the cities or private models like Uber. This may result in different vehicle types for different purposes and could change the urban distribution. Once AVs are common, this presented group mechanism may be useful for managing future mobility.

In theory, vehicle fleets called formations of formations are possible. Due to the focus on keeping the individuality of the vehicles, this was not rethought for the investigations in this thesis. Thus a research question could be when, why and how to merge and cluster groups. This research direction could make use of database knowledge and machine learning.

Inspired by the reservation system [89], the traffic lights give guarantees to vehicle groups. Thus, traffic lights benefit from a grouping concept to synchronize information and react better to possible traffic demand. The research question would deal with the decentralization and self-organizing of traffic management including the technology of vehicular and infrastructure communication. This could join the domains of agents or organic computing for organization (as in works by Pohlmann [274], Müller-Schoer and Hähner [279]), and dynamic traffic management for providing the expert knowledge and rethinking models towards information sharing, which is done with communication, for example by the postal service. Future investigation can deal with information from the same and/or opposite vehicles what information could it be desirable to exchange through communication?

Malicious behavior or competitive aims are not covered by this thesis. The idea of this work was to assume that every vehicle wants to maximize the utility. In the context of autonomous vehicles, where getting quickly from A to B is the goal, it idealized that all vehicles are regarded as equal. The research question could define and formulate cases of traffic participants with bad intentions in urban traffic, or when different priorities are desired. Works of Marin Lujak [42, 236] deal with high priority vehicles for emergencies like ambulances, firemen, or the police. For future research, security, trust and game-theoretic research domains could contribute and extend to the future mobility. Traditional game theory is effective, but cannot explain the result of collective agent decisions. Interaction among agents determines their cooperation. Future research could create a family of sociograms using small-world networks. In social-learning algorithms agents can learn from other peer agents with the help of automating the negotiation and cooperation strategies.

The AVGF algorithm of this thesis covers the similarities of vehicles including their route. But an open research topic is route planning for vehicle groups. The challenge is that members of a group can agree on a route choice, e.g., whether the economically optimal route is taken, or turning left should be avoided when possible. Relevant research on this topic is being done by a colleague Sophie L. Dennisen [82]. Furthermore, best algorithms can be evaluated for finding the best group route. The shortest path is a planning algorithm for routing optimization and is relevant because best routes must be calculated within a few seconds and the algorithm must be repeatedly applied to find the current best remaining route to the destination. Therefore, it seems more useful for the use cases done in the MATI simulator.

The AVGF approach is used for the problem through a microscopic user view. For future research an auction algorithm with AVG is proposed. In order to fulfill global and individual utilities, this provides a solution for the optimization problem. Considering that negotiations are made between autonomous vehicles and software-responding traffic-dependent lights, guarantees are given by spatial-temporal pairs on the sections and intersections on the itinerary of each AV. In the case that the itinerary is not possible due to traffic jams on the road, AVs search for the second best itinerary until the infrastructure confirms their availability, as described by `Dresner and Stone` [89].

Currently, multi-agent systems are implemented using multi-threading concepts. With the increasing availability of clusters, parallel computers, and clouds, there is a need to redesign the construction of agent-based systems, leveraging the underlying processors. A venue for future research is done with works by `Aschermann et al.` [8] to improve the agent simulations to make them more scalable.

There are still issues to be addressed, however, the results of this research on autonomous vehicle groups have so far been extremely encouraging. With continued research, it is feasible that cooperative autonomous vehicles will coexist alongside human drivers in the not-too-distant future.

In the foreseeable future, the technique of autonomous driving will be available in almost every vehicle. Therefore, the car industry will interconnect closely with information technology like telecommunication services, Apps for costumer service or entertainment. A lot of data will be collected and cars will interlink for data communication. Risks exist in data misuse and surveillance. Many things need to change or adapt in order to reap the benefits of time and fuel savings, as well as less accidents than human drivers. The acceptance and fun factor of non-autonomous driving, the inter-human psychology to go the step into the future and leave behind history, prevailing legal norms for traffic offenders and cases of damage need to be specified, as well as data security. New infrastructure will also follow up, of no traffic lights and signs necessary through communication, or standards will evolve to relieve traffic chaos. Concluding that the presented concepts of this work will find their way from simulation into application in the next 20 years, this thesis provides a small step and a building block to this development.

Appendices

It has become appallingly obvious that our technology has exceeded our humanity.
Albert Einstein (1879-1955)

Appendix A

Car Following Models

A.1 Nagel-Schreckenberg Model (1992)

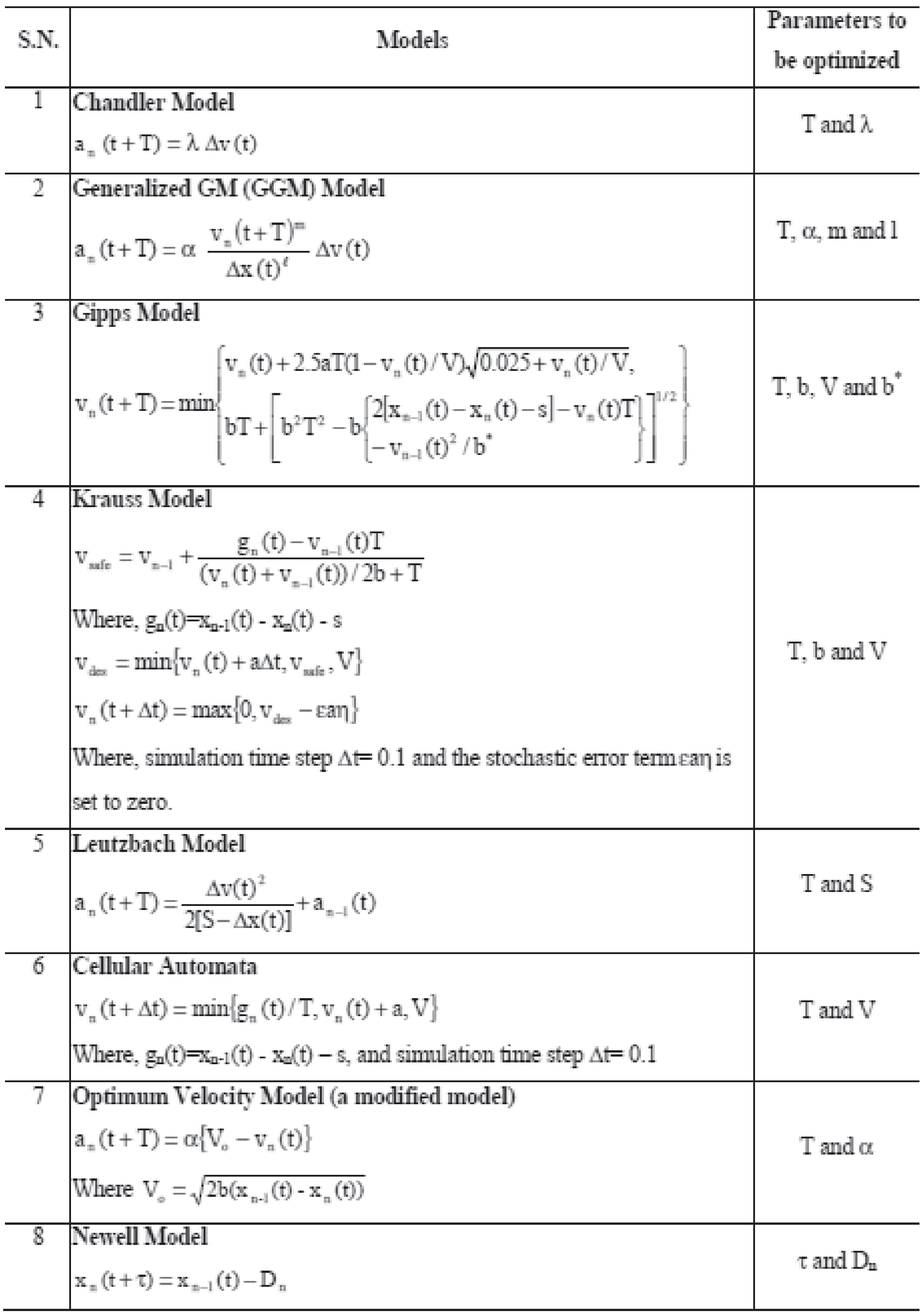
Cellular Automata aim to reproduce traffic flows and are another method of microscopic traffic simulations. Basically cells divide the space into segments and the state is updated depending on a set of update rules and the state of neighboring cells.

The most popular cellular automata model (CA) is the `Nagel and Schreckenberg` [259] model which is discrete in space and time. It describes vehicles on cells with their states of their presence or absence and speed. Therefore it reduces the computational complexity, but is still able to mimic drivers' reaction in response to the environment. It describes the traffic system as a lattice of cells of a equal size and similar to psycho-physical models set of rules provide control the movements of the vehicles from cell to cell. Cell sizes are chosen that it can contain one vehicle which can move to the next cell in one time step Δt. The velocity is expressed as the number of cells per time step usually $\Delta t = 1$ cf. [154].

The update of the positions of the vehicles is based on four rules.

1. **Acceleration**: $v_i(t+1) = min(v^{max}, v_i(t))$
2. **Breaking**: $v_i(t+1) = min(v_i(t), \Delta x_i(t))$
3. **Randomization**: $v_i(t+1) = rand(0 \ldots (v^{max} - 1))$ with the probability ρ
4. **Moving**: $x_i(t+1) = x_i(t) + v_i(t+1)$

In the spatial units speed adaptation is done depending on the headway gap and speed of the car ahead and on a probabilistic deceleration rate. Implementing cellular automata can be done efficiently with simple rules of the Nagel Schreckenberg model and therefore are suitable for large-scale traffic flow simulations considering that update rules are strictly local.



S.N.	Models	Parameters to be optimized
1	**Chandler Model** $a_n(t+T) = \lambda \, \Delta v(t)$	T and λ
2	**Generalized GM (GGM) Model** $a_n(t+T) = \alpha \dfrac{v_n(t+T)^m}{\Delta x(t)^\ell} \Delta v(t)$	T, α, m and l
3	**Gipps Model** $v_n(t+T) = \min \left\lbrace \begin{array}{l} v_n(t) + 2.5aT(1 - v_n(t)/V)\sqrt{0.025 + v_n(t)/V}, \\ bT + \left[b^2T^2 - b \left\lbrace \begin{array}{l} 2[x_{n-1}(t) - x_n(t) - s] - v_n(t)T \\ - v_{n-1}(t)^2 / b^* \end{array} \right\rbrace \right]^{1/2} \end{array} \right\rbrace$	T, b, V and b*
4	**Krauss Model** $v_{safe} = v_{n-1} + \dfrac{g_n(t) - v_{n-1}(t)T}{(v_n(t) + v_{n-1}(t))/2b + T}$ Where, $g_n(t)=x_{n-1}(t) - x_n(t) - s$ $v_{des} = \min\lbrace v_n(t) + a\Delta t, v_{safe}, V\rbrace$ $v_n(t+\Delta t) = \max\lbrace 0, v_{des} - \varepsilon a \eta\rbrace$ Where, simulation time step $\Delta t= 0.1$ and the stochastic error term $\varepsilon a \eta$ is set to zero.	T, b and V
5	**Leutzbach Model** $a_n(t+T) = \dfrac{\Delta v(t)^2}{2[S - \Delta x(t)]} + a_{n-1}(t)$	T and S
6	**Cellular Automata** $v_n(t+\Delta t) = \min\lbrace g_n(t)/T, v_n(t) + a, V\rbrace$ Where, $g_n(t)=x_{n-1}(t) - x_n(t) - s$, and simulation time step $\Delta t= 0.1$	T and V
7	**Optimum Velocity Model (a modified model)** $a_n(t+T) = \alpha\lbrace V_o - v_n(t)\rbrace$ Where $V_o = \sqrt{2b(x_{n-1}(t) - x_n(t))}$	T and α
8	**Newell Model** $x_n(t+\tau) = x_{n-1}(t) - D_n$	τ and D_n

Figure A.1: Table with Car Following Models [284].

Figure A.2 illustrates a single lane road which is a typical application of the `Nagel and Schreckenberg` model. The dynamics are represented by the notation $action_i : v_i(t) \rightarrow v_i(t+1)$ which is for example $Acc_i : 3 \rightarrow 4$ indicating that vehicle i is in the acceleration phase between time t and $t+1$ and its speed increase from 3 to $4m/s$.

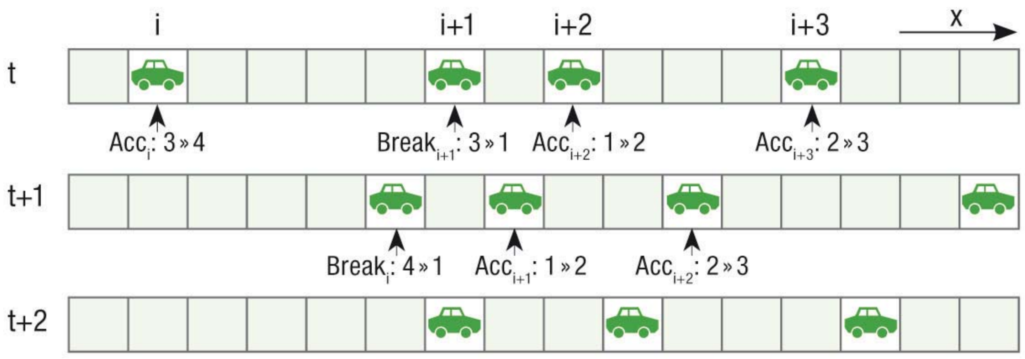


Figure A.2: Nagel-Schreckenberg cellular automaton model (cf. [154] p. 121).

At time t there are four vehicles in the acceleration phase. In the next time step, the vehicle i previously moved three cells and accelerates to increase its speed to move four cells when it has at least that number of free cells ahead. Then at time step $i+1$ the vehicle i is very close to the front vehicle $i+1$ and must break to reduce its speed to one cell which is free available ahead. The model has a critical number of vehicles which limits the maximum speed, but it may produce sudden and unrealistic behavior due to the synchronous movements at the next time step. That is the case that if vehicle i would know that the front vehicle $i+1$ was already breaking, it could have avoided accelerating and preemptively even decelerated. But due to the decision at the same time the information is not available. This can even result in a domino effect because the action of the leading vehicle depends on its own leading vehicle. Anyhow, the `Nagel and Schreckenberg` model efficiently models vehicular traffic and is used by the TRANSIMS [218] traffic simulator.

A.2 Kinematic Distance Model

The kinematic distance model is based on a simple equation of motion. Considering the state of the predecessor, the own acceleration of a vehicle is calculated such that collisions are avoided.

$$a_{n+1} = a_n + (D_x - \Delta x) \cdot \frac{2}{t_c^2} + (v_n - v_{n+1}) \cdot \frac{2}{t_c} \tag{A.1}$$

with

a_n	acceleration of vehicle n (predecessor)	$[m/s^2]$
D_x	current gross distance	$[m]$
Δx	minimum gross distance	$[m]$
t_c	time-to-collision[1]	$[s]$
v_n	speed of predecessor	$[m/s]$
v_{n+1}	speed of the following vehicle	$[m/s]$

Note that the index n represents the vehicle ahead and $n+1$ the subsequent following vehicle, see 2.5. As an example imagine a vehicle that approaches a slow vehicle ahead.

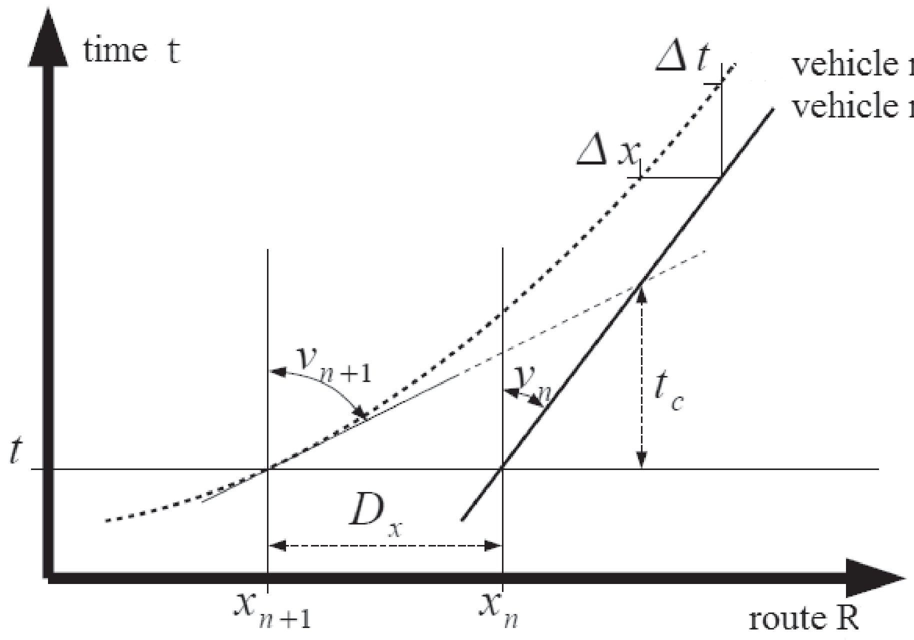


Figure A.3: Two Vehicles in a Route-Time-Diagram.

This kinematic model results in a behavior illustrated in A.3. The x-coordinate axis plots the route R and the y-coordinate axis measures the time t. The vehicle ahead n is represented by a continuous line and the following vehicle $n+1$ by a broken line. By the instant of time t the vehicle $n+1$ has more speed than the predecessor n and therefore the distance reduces with progressing time. In this example the predecessor n drives with constant speed, thus, having an acceleration of $a_n = 0$. The equation A.1 results in a negative acceleration for the vehicle $n+1$ when due to the decreasing distance D_x the term $(D_x - \Delta x) \cdot 2 \cdot t_c^{-2}$ carries less weight than the term $(v_n - v_{n+1}) \cdot 2 \cdot t_c^{-1}$ which includes the speed difference $(v_n - v_{n+1} < 0)$. The vehicle $n+1$ decelerates until it reaches the same speed $v_{n+1} = v_n$ like his predecessor n and can follow with a minimum distance Δx.

A.3 Model by Pipes (1953)

In 1953 simple car following model [273] was developed by the physicist `Louis Albert Pipes` which gives the base for more complex models. The main idea is that all vehicles are observing of safety distances to their predecessor. Its origin comes from a Californian Court Case [357] and manifested by the Californian Legislative Information [224]. It was documented that every $10mph$ ($16km/h$) speed there should be a distance of one vehicle length 15 feet ($4,572m$). The rule was to leave one car length for every 10 miles per hour ($"mph", 16km/h$) of speed. For example, if following the car ahead at $30mph$ (about $50km/h$), there need to be three car lengths.

$$x_n(t) - x_{n+1}(t) = \Delta x_{safe} + \tau v_{n+1}(t) \tag{A.2}$$

with

Δx_{safe}	safety distance	$[m]$
τ	time scale of the relaxation process	$[s]$

In this equation τ is a measure for sensitivity. In order to ensure a safe distance to the predecessor, every vehicle driver tries to adjust its speed to the vehicle ahead. This simple model implies that the desired speed corresponds to the speed of the predecessor. Therefore, differentiating the equation A.2 into the equation of motion results in:

$$\dot{v}_{n+1}(t) = \frac{V_{des} - v_{n+1}(t)}{\tau} = \frac{v_n(t) - v_{n+1}(t)}{\tau} \tag{A.3}$$

V_{des}	desired speed	$[m/s]$

Through this interaction of neighboring vehicle driver an equilibrium is regulated when all vehicles drive with the same speed behind each other.

A.4 GHR Model (1961)

The model by `Gazis, Herman and Rothery (GHR)`[125] is based on the approach of `Pipes` A.3. The model aims for the compliance with safety distances of different vehicles in a platoon. The basic idea is that the driver reacts with a response proportional to a given stimulus.

$$response(t+T) = \alpha \cdot stimulus(t) \tag{A.4}$$

α	sensitivity factor	$[1/s]$

This principle is used in many other car following models, whereas $/alpha$ represents the sensitivity factor and depending on the model it can be chosen differently. Often it depends on the position, speed, acceleration of the involved vehicles. The time T is the reaction time which passes until there is a response to the stimuli. In this model the stimuli is equivalent to the speed difference to the vehicle ahead.

$$stimuli(t) = \Delta v(t) = v_n(t) - v_{n+1}(t) \tag{A.5}$$

The index $n+1$ also indicates the following vehicle likewise the notation in 2.2.4. In this model, the reaction relates to the acceleration of the driver. Therefore, when approaching the vehicle ahead ($\Delta v(t) < 0$), the following vehicle $n+1$ reacts with deceleration.

In times of commencement of the model, the factor of sensitivity α was constant. Consequential the acceleration $\dot{v}$ is described in the equation:

$$\dot{v}_{n+1}(t+T) = \alpha \cdot [v_n(t) - v_{n+1}(t)] \tag{A.6}$$

The resulting model, also known as linear car following model, consists of adding the reaction time into the equation A.3 of `Pipes` with $\alpha = \frac{1}{\tau}$. This severely reduced model will not return realistic results because the reaction of the driver is only influenced by the speed difference. Neither the distance to the predecessor nor the exact amount of speeds have influence on the reaction. Therefore the equation of `Pipes` was extended such that distances and amount of speed have an influence on the reaction in this equation, also known as non-linear car following model:

$$\dot{v}_{n+1}(t+T) = \alpha \cdot \frac{v_{n+1}^{m}(t+T)}{[x_n(t) - x_{n+1}(t)]^l} \cdot [v_n(t) - v_{n+1}(t)] \tag{A.7}$$

Once again the sensitivity factor α is constant, as well as the model-specific parameter m and l. The current position of a vehicle at the time t is specified with $x(t)$, hence the distance between two vehicles is $[x_n(t) - x_{n+1}(t)] = \Delta x$.

A.5 Model by Wiedemann (1974)

So far, the presented models calculate the behavior of a vehicle with the assumption that the parameter like distances and speed differences are known numerically exact. In reality the driver can only estimate the actual traffic state. A change of distances or speed difference can just be observed if the perception is above a certain threshold [248, 344].

The model of `Wiedemann` [377] is a representation of reality to include the effects of perception and is known as the *psycho-physical model*. This model is used in the commercial traffic simulation software $PTV-VISSIM^{TM}$ and in a SLX-based [310] psychophysical vehicle-following simulation mode. `Wiedemann` describes in his work the processes which influence the estimation of the speed difference.

The driver of a vehicle tries to observe another vehicle to estimate the speed difference and the change of it.

A change of the angle α can be interpreted as a measure for the speed difference formulated by:

$$\frac{d\alpha}{dt} = \frac{c \cdot \Delta v}{\Delta x^2} \tag{A.8}$$

This angle A.8 depends on the width of an object c which is constant and the distance Δx between the object and the observer. In reverse, the speed difference equates to zero and therefore also the change of the angle $\frac{d\alpha}{dt} = 0$ when the driver wants to keep constantly the same distance to his predecessor.

All speed differences need to be reacted on whenever those are dissimilar to zero and therefore all angle speeds as well. But the human eye cannot react on very small angle speeds, only if a certain perception threshold S passed, the angle change and therefore the speed difference is noticeable[2].

The model of `Wiedemann` is divided into four different driving states which are:

- **Desire**: not influenced by other traffic participants
- **Follow**: unconsciously influenced by the predecessor (speed adjustment by accelerator, no breaking)
- **Active Break**: consciously influenced by the predecessor (active breaking)
- **Maximum Break**: consciously influenced by the predecessor (maximum breaking for avoiding a collision)

In every time step of the simulation, the driver is in one of those discrete states. Not before the driver-specific thresholds of perception are exceeded, a transfer between the states happens.

The vehicle in state **Desire** accelerates until it reaches its desired speed. The distance to the predecessor is large enough not to influence this driving behavior. Once the desired speed is reached, it continues driving with constant speed. In case the speed is bigger than the speed of the predecessor, the distance reduces. This state continues until the threshold of perception between large distances $25 * (\Delta v)^0, 5$ and $75 * (\Delta v)^0, 5$ is reached.

The vehicle (=driver) reaches the state **Active Break** when it reacts with breaking due to reduced distance to the predecessor. The speed difference is adjusted. When the speed is reduced to the upper limit of $35m(\Delta x)$, then the next state is **Follow**. The speed is regulated unconsciously just by acceleration, not by breaking. With unchanged situation of the speed of the predecessor, the vehicle stays in this state. The lower limit is $20m(\Delta x)$ as the minimum following distance with the same speed. Additionally this state is limited by a perception threshold of speed differences which varies even for the same driver.

For unpredictable situation of a collision avoidance, the state is **Maximum Break** is initiated, if the gross distance Δx is smaller than the length of the predecessor. The modeled safety is the sum of the length of the predecessor and the desired distance when completely stopped.

A.6 Optimal Velocity Model (1995)

In 1995 the Optimal Velocity Model (OVM) was introduced by Bando et al. [18] for traffic congestion and refined with delays in [17] and describes the following features:

[2]The threshold of perception S differs for each individual and lies around $3 * 10^-4$ to $10 * 10^-4 rad/s$

- A vehicle will keep the maximum speed with enough the distance to the next car.
- A vehicle tries to run with optimal velocity determined by the distance to the next car.

This model calculates an appropriate speed with respecting only the vehicle driving ahead and not the speed difference between the vehicles. The idea is that respecting the distance to the front vehicle and using the 'optimal velocity function' $V(\Delta x)$ the desired speed can be determined. The acceleration for the time t is calculated by the difference between current and desired speed.

$$\dot{v}_n(t) = \alpha \cdot [V(\Delta x) - v_n(t)] \tag{A.9}$$

	α	driver specific sensitivity factor	$[1/s]$
with	Δx	gross distance to the front vehicle	$[m]$
	$V(\Delta x)$	Optimal Velocity function	$[m/s]$

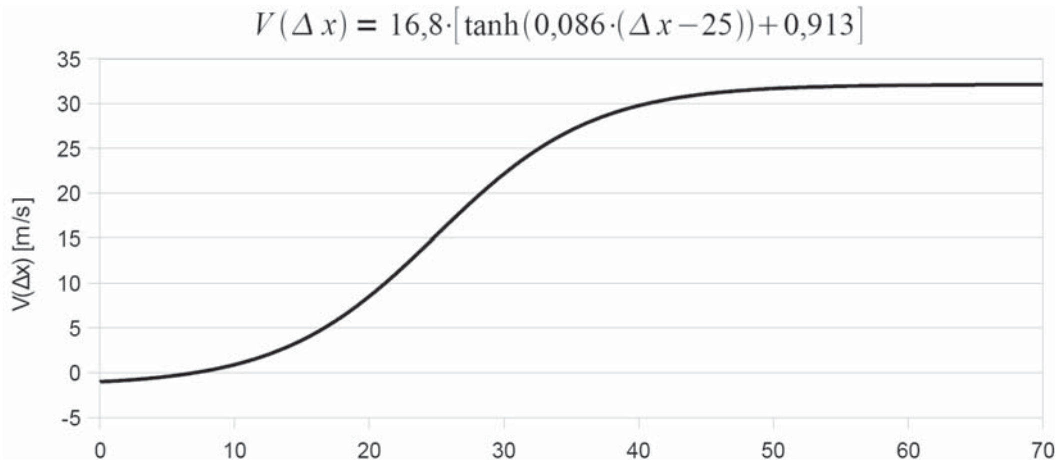


Figure A.4: Optimal Velocity Function from empirical data by [18].

In A.4 the function $V(\Delta x)$ is monotonically increasing and for $\Delta x \to \infty$ it is limited by the maximum speed. Whereas in the beginning the resulting speed strongly increases for lengthening distances, the vehicle later will reach its limits. Thus, the speed increasing slows down until it reaches its maximal speed. There is a inflection point within the course of the function:

$$V(\Delta x) = V_0 \cdot [tanhm \cdot (\Delta x - b_f) - tanhm \cdot (d - b_f)] \tag{A.10}$$

	d	gross distance at standstill	$[m]$
with	b_f	gross distance on the inflection point of the function	$[m]$
	m	gradient at inflection point	$[1/m]$
	V_0	Maximum speed minus speed at inflection point	$[m/s]$

The illustrated course of the function is based on the term:

$$V(\Delta x) = 16{,}8 \cdot [tanh(0{,}086 \cdot (\Delta x - 25)) + 0{,}913] \tag{A.11}$$

The function term specified by Bando et al. which is empiric data obtained on a Japanese highway. Remarkable is that there is negative speed for distances Δx smaller than $7m$ which is identical to the gross distance at standstill. In the presented function there is the maximum speed of 32,13m/s which is equivalent to 115,67km/h.

A.7 Intelligent Driver Model (2000)

In 2000 the Intelligent Driver Model (IDM) originates by Treiber et al. [243]. It is a continuous and deterministic model which represents the driving behavior of a driver-vehicle unit by calculating the acceleration with an equation. It is similar to earlier models, because the parameters of interest are the distance and the speed difference to the front vehicle. The idea of the model is to aim for realistic simulation properties and an easy modeling of different driving styles and vehicle properties i.e. less parameter than the model of Wiedemann A.5. For example the desired speed, acceleration and deceleration are less of a bus or truck than the parameter of a sports car which should be included in the model.

$$\dot{v}_a = a \cdot \left(\underbrace{1 - \left(\frac{v_a}{v_0}\right)^{\delta}}_{FreeDrive} \underbrace{- \left(\frac{s^*(v_a, \Delta v_a)}{s_a}\right)^2}_{Interaction} \right) \tag{A.12}$$

with

with	a	maximal acceleration of vehicle	$[m/s^2]$
	v_a	current speed	$[m/s]$
	v_0	desired speed	$[m/s]$
	δ	exponent for acceleration (often $\delta = 4$)	$[m/s]$
	Δv_a	speed difference to front vehicle	$[m/s]$
	s_a	net distance to front vehicle[3]	$[m]$
	$s^*(v_a, \Delta v_a)$	actual desired distance	$[m]$

The equation A.12 consists of first an acceleration part on a free route and second a deceleration interdependency through the interaction with other traffic participants. In the latter the effective net distance s_a is set into proportion with its *actual desired distance*:

$$s^*(v_a, \Delta v_a) = s_{0+v_aT+\frac{v_a\Delta v_a}{2\sqrt{ab}}} \tag{A.13}$$

with	s_0	minimal distance to front vehicle	$[m]$
	T	safety distance by time	$[s]$
	b	comfortable deceleration delay	$[m/s^2]$

Thus, there is a continuous and deterministic dependency of the acceleration of local determining factors. The model does without discrete ranges of driving dynamics which is why no complex if-then decision logics is required like in the model of Wiedemann A.5.

Beauty is more important in computing than anywhere else in technology because software is so complicated. Beauty is the ultimate defense against complexity. - D. Gelernter ("Machine Beauty", Basic Books, 1998)

Appendix B

Existing Agent-Oriented Simulation Platforms

In the article of `Nikolai and Madey` [262] 53 simulation platforms are mentioned with categories of used programming language, operating system, license type as well as application field. Subcategories of common characteristics are listed, so that user with certain preferences can select their simulation platforms. The authors do not provide an evaluation of the platforms.

The survey of `Allan` [6] provides more detail for the 40 simulation platforms. In form of text `Allan` gives a general description of each software and selects user groups for certain characteristics which bundles the interests. This technical report also shows which kind of simulation can be done with the simulation software. The author gives hints for the technical use and their interrelationship, for example about the origin of the simulation platform. On this base a selection can be grounded with objective reasons.

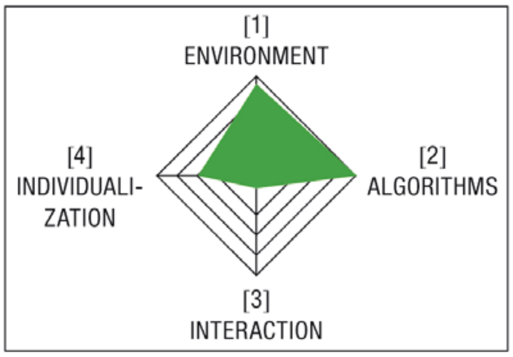
Other selection criteria was personal experience with simulation platforms and preference for Java-based programs, talks with researchers with experience in agent-oriented simulation platforms of the Technical University of Clausthal and agent focused conferences, and no use of very specialized or old and inactive software. This results in eight agent-oriented simulation platforms: (1) AnyLogic, (2) JADE, (3) Jason, (4) MASON, (5) MATSim, (6) NetLogo, (7) Repast Simphony, and (8) SeSam.

This is the focus of the investigation analysis of agent-oriented simulation systems in this Section. Ideally the software can represent the following content and processes for an objective evaluation of the requirements requirements for Environment, Algorithms, Interaction, Individualization (EAII):

1. **Environment:** The modeling of an environment in form of road networks (streets, intersections) and street elements (traffic lights). The infrastructure is the base for a traffic scenario.

2. **Algorithms:** Vehicles should be implemented with car following models to avoid accidents through their behavior and ideally with algorithms to calculate routes. Mathematical models are provided.
3. **Interaction:** The interaction between vehicles (V2V), as well as to the infrastructure (V2I) should be supported by the software. Communication is implemented.
4. **Individualization:** Vehicles should be represented and programmed individually. Agent concept is useful.

Especially the information exchange can improve driver-independent (autonomous) systems and the traffic throughput and on that base also coordination can take place.



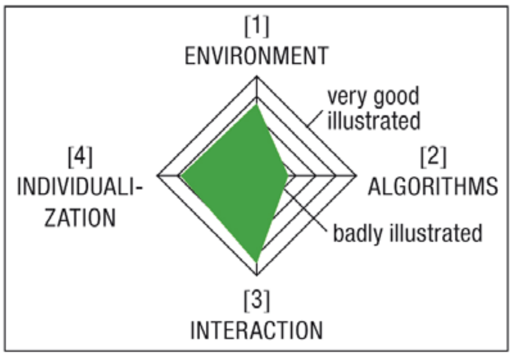


Figure B.1: Strengths and Weaknesses of Transport Simulation Systems versus agent-oriented simulation platforms. [cf. Masterthesis `Eichhorn` [97]]

Here two kinds of software need to be differentiated which could fulfill those EAII requirements and are visualized exemplary in Figure B.1:

- **specialized transport simulation systems** like AIMSUN, SUMO, VISSIM, AVENUE, PARAMICS, MITSIMLab, DRACULA, Dynameq, DynaMIT, or METNET listed in `Barceló` [21] can simulate complex road networks. As an exemplary overview in Figure B.1 (5) shows the strengths and weaknesses of the transport simulation system regarding the requirements described above. As a general rule such programs can model detailed environments. Algorithms for vehicles are implemented with car following models like time-continuous models, iterated maps, cellular automata, behavioral components, discrete decision models and road network support for imports and modeling as well as the output of travel times, fuel consumption model, detectors and spatio-temporal contours. Whereas possibilities for interaction are usually very limited, as well as the individual programming of the vehicles.
- **more general use agent-oriented simulation platforms** like AnyLogic, JADE, Jason, MASON, MATSim, NetLogo, Repast Simphony, or SeSam. As also seen in Figure B.1 (6) the strengths and weaknesses of agent-oriented simulation platforms are distributed different from the

Table B.1: Agent-oriented simulation platforms with efficiency criteria.

Number	Agent Simulation	Efficiency tested Boids Flock
1	Anylogic	3: around 1120 agents
2	JADE	-: Party 1000 agents
3	Jason	-: Game-of-life several 1000 agents
4	MASON	1: over 3600 agents
5	MATSIM	-: Zurich innercity 5000 agents
6	NetLogo	2: about 1900 agents
7	Repast Simphony	4: 150 agents
8	SeSAm	5: 120-130 agents

transport simulation systems. The environment is generally just half supported compared to the transport simulation, algorithms are generally not pre-implemented. But the strengths are in the individualization of the agents with BDI models and in interaction.

The focus in this section is on the agent-oriented simulation with adding aspects of the transportation requirements of the environment and implementation of algorithms.

B.1 Efficiency of Agent Simulation

Note on efficiency: the agent-oriented simulation platforms are too different to compare. Therefore, it is not possible to get durations and time for a certain reference simulation for the evaluation purpose. But an example simulation is *Boids Flocking Simulation* which researches bird swarm behavior is executed with 2000 simulation steps with a limit of 100 seconds for calculation and searched is the number of agents. The results are listed in Table B.1 for the different agent simulation.

JADE, Jason and MATSim are without an efficiency statement, they did not have the simulation models for testing when executed in 2012.

JADE is a special case because by default there is not graphical representation of the simulation. The example simulation ”Party” could be done with 1000 agents without problems and in a very short time.

Jason calculated a example simulation ”Game of life” with several thousand agents.

MATSim functions different, first the simulation is calculated and then the results of the calculation are saved. With an additional program the simulation results can be visualized. The tutorial uses a scenario of the innercity of Zurich, Switzerland and can calculate 5000 agents without problem.

More information about the efficiency test is provided in the master thesis by `Eichhorn` [97].

Version	6.8.1 compared (current 7.2.0)
What is AnyLogic?	AnyLogic is a multi-method simulation tool and supports three different modeling techniques: discrete event, systems dynamics and agent-based.
Developers	XJ Technologies was the developing company of AnyLogic and founded in 1992. For the first time AnyLogic was presented on the Winter Simulation conference in year 2000. The headquarters are in St. Peterburg Russia.
Properties	AnyLogic can combine its three modeling techniques and supports 2D and 3D. It has an export function of the models as an Java applet.
Features	Changes can be made during a running simulation, the simulation runs time-based. Pedestrians, trains, and traffic are supported with libraries for supporting the models.
Further information	Articles and publication are provided on the homepage for different industries, application area and with simulation methods. Three books are available: most notable the book "as a bible" by Dr. Andrei Borshev [50] the CEO and the (free download) "AnyLogic 7 in three days" book by Ilya `Grigoryev` [142] and a Business Dynamics book by John `Sterman` [328].
Documentation	Lots of example models, but no documentation. Partly tutorials are videos.
Licence	30 days free trail, commercial
Operating System	windows/MacOS/Linux
URL	`/http://www.anylogic.com`

Table B.2: Overview Summary of AnyLogic.

B.2 AnyLogic

B.2.1 Installation

The AnyLogic Company provides for all operating systems professional, advanced, researchers/educational and free PLE (Personal License Edition) downloads. System requirements and JRE (Java Runtime Environment)and Java-Plugin are mandatory. After installation Anylogic is started with the .exe file and then the software needs to be activated with a license key, then use AnyLogic.

B.2.2 Advantages

AnyLogic has lots of information on their website to attract consumers of different industries and researchers alike. AnyLogic has a huge sample model library and also the video tutorials help to get started with the program. Comprehensive libraries support the modeling and AnyLogic can combine three different modeling techniques. Projects can be saved as an applet with the export function and demonstrated in the browser.

B.2.3 Disadvantages

There are many sample models, but the corresponding documentation is missing. The functions of the software are large and therefore the program has a high complexity. A useful combination of the modeling techniques is partly very difficult. AnyLogic is the only commercial simulation platform which is a downside.

B.2.4 Conclusion

AnyLogic offers a high degree of functions and three different modeling techniques, besides agent models, discrete and system dynamics. Therefore lots of simulation scenarios can be modeled. In the ranking of modeling a simulation environment and the graphical user interface with supporting libraries AnyLogic gets the best results. For modeling with agents the state diagrams are useful, although the degree of function for the interaction are limited compared to the alternatives of Jason and JADE. In older versions there were problems with the modeling of individual agent behavior combined with traffic simulation. But the traffic library and the agent-based modeling technique are a promising alternative for the future to simulate in cooperative traffic.

B.3 JADE

B.3.1 Installation

On the JADE homepage there is a JADE-all-x.x.x.zip provided which contains all relevant data. After unpacking the files, JADE can be started with a command in the console. The administration tool provides a detailed instruction.

As a programming environment JADE is not very restrictive. It has a minimalistic GUI which can be displayed when starting MAS, but generally there is no user interface. The GUI shows active agents and offers options to control and observe the agents.

Version	4.2.0 compared (current 4.3.3)
What is JADE?	JADE is an opensource middleware which supports the development of agent-based applications with peer-to-peer technology.
Developers	February 2000 JADE was developed by Telecom Italia Lab. Since 2003 there exists a JADE Board consisting of 5 companies which control the management of the JADE project.
Properties	JADE offers a runtime environment for MAS and a library for agent-oriented programming. There are tools to support debugging and deployment phases.
Features	The agent platform can be distributed on several machines. The configuration can be controlled by "remote GUI". JADE is FIPA conform, the standard for communication for agents.
Further reading	JADE has publications on its website and the standard work is the book "Developing Multi-Agent Systems with JADE" [36].
Documentation	Next to the book, there is a tutorial for administration which describes the architecture, how JADE is started and the creation of a distributed platform and how JADE can run on several machines simultaneously. Another programming tutorial is presented with a book exchange scenario. More links like the API documentation are available.
Licence	open source
Operating System	windows/MacOS/Linux Java 5.0 or higher required
URL	`http://jade.tilab.com`

Table B.3: Overview Summary of JADE.

B.3.2 Advantages

JADE is very useful for complex communication. JADE supports distributed system topology and peer-to-peer networking. As a middle-ware it has many combination possibilities to other programs.

B.3.3 Disadvantages

As a standalone simulation platform JADE is not suitable for cooperative traffic scenarios. The GUI is very minimalistic and only offers limited tools for administration. JADE does not offer any support for modeling the infrastructure environment. For the use of JADE - especially for combination of other platforms advanced JAVA knowledge is necessary.

B.3.4 Conclusion

In general JADE is not used as a standalone simulation platform, but a middleware. Just in combination with other programs JADE can contribute with its communication strengths especially. But also the architecture support to distribute the simulation in a network of connected computer. As a standalone JADE is not helpful for cooperative traffic scenarios.

JADE can be used in combination for traffic scenarios. Two own publications JRep [138] and ATSim [65] (which is later also described in Section 4.3.4) successfully combined JADE with other simulation environments.

JRep [138] connects the simulation platforms Repast S (also described as a standalone agent platform in Subsection B.8) and JADE. The aim of this JRep project was to join the benefits of both platforms into a more powerful combination. In JRep Repast S is used for the control of the simulation and the analyzing and evaluation of the data, as well as the representation of the global scenario. JADE is used to model individual agent behavior and to assure the communication between the agents. A new architecture is proposed where JADE agents can be integrated into Repast S environment. With an example the scalability and the performance of the new JRep Simulation was tested in a coin-flip scenario and it shows that JRep has good results with 10000 agents.

B.4 Jason

B.4.1 Installation

For starting Jason, the .zip file needs to be downloaded from the homepage and unzipped and then the .exe file starts the functionalities of Jason.

B.4.2 Advantages

Jason uses AgentSpeak as a programming language which was designed for the development of complex agent behavior. The Mind-Inspector-Tool is very

Version	1.3.6 compared (current 1.4.2)
What is Jason?	Jason is an agent interpreter for the extended version of the programming language AgentSpeak. It implements an operational semantic of the language and offers a platform to develop multi-agent systems.
Developers	Mainly Jason is developed by Jomi F. Hübner (Department of Automation and Systems Engineering, Federal University of Santa Catarina, Brazil) and Rafael H. Bordini (Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brazil).
Properties	Jason is based on the BDI agent model and supports the environment creation with Java. The development environment is integrated with jEdit or Eclipse Plugin. The environment contains a ”mind inspector” during runtime the agent behavior can be observed in detail and is useful for error handling.
Features	It supports organizational agent hierarchies with the extension Moise+ and a sample is a soccer team, Jason uses the FIPA standard for communication of agents, with the use of Sci or JADE it can be distributed over a network, a multi-agent contest is also organized by the TU Clausthal, Germany.
Further reading	The book ”Programming Multi-Agent Systems in AgentSpeak using Jason ” [49] is very useful to get into Jason. The exercises are sophisticated.
Documentation	On the homepage is a getting started tutorial, publication and FAQ are also available with a good API documentation.
Licence	open source
Operating System	windows/MacOS/Linux
URL	`http://jason.sourceforge.net`

Table B.4: Overview Summary of Jason.

useful to understand the agent behavior and to solve programming errata. The extension Moise+ offers organizational functionalities for group behavior and CArtAgO helps to simplify the environment, combined the extensions are called JaCaMo.

B.4.3 Disadvantages

For using Jason advanced programming knowledge of Java and AgentSpeak is required. Getting started with the program is difficult and the exercises are changes in already complete sample models.

B.4.4 Conclusion

Jason is a good choice for developing scenarios with interacting agents. With the implementation of the BDI model as an agent model Jason provides an interesting alternative to the other simulation platforms. When modeling an environment more Jason support could help. At present there are no traffic simulation as samples where the full potential of Jason is not fully exploited. Jason is difficult to get started, but once with the knowledge it is a powerful agent simulation.

B.5 MASON: Multi-Agent Simulator Of Neighborhoods/Networks

B.5.1 Installation

For using the functionalities of MASON, the mason.zip file can be downloaded form the MASON homepage. In this package is a folder with the name jar and in that is the file mason.xx.jar which needs to be integrated into a project like Eclipse or Netbeans. Alternatively the MASON project can be started via the console.

B.5.2 Advantages

The tutorial is for advanced Java developers and is very comprehensive with 14 parts. MASON supports Agent-to-X communication. Another big advantage is the performance which states in [283] to be the fastest compared to NetLogo, Repast, or Swarm. The MASON project is programmed with Eclipse and can be extended.

B.5.3 Disadvantages

Compared to MATSim described in Subsection B.6 is the introduction to MASON more difficult, because the understanding of required classes needs time. The GeoMason Project Gridlock offers no documentation up to date. The

Version	16 compared (current 19)
What is MASON?	MASON is a Java-based event-oriented multi-agent simulation toolkit.
Developers	MASON is a joint project of two departments of the George Mason University: Computer Science and Center for Social Complexity. The university is located in Fairfax, Virginia, USA.
Properties	MASON contains a modeling library and a bundle of visualization tools to visualize in 2D and 3D.
Features	MASON was developed with the goal to give best support for computationally intensive models with many agents for many iterations. The simulation model can be executed without visualization. The developers state that without visualization the simulation can support up to one million agents. With the visualization the amount of agent reduces depending on the computer memory. The extension GeoMason offers to use GIS.
Further reading	The paper [237] describes the motivation of the development of MASON and the used architecture. Application examples are presented and explained. More paper are available on the website.
Documentation	MASON offers lot of executable example models, but not of the traffic area. Moreover there is a 300 pages pdf manual and a tutorial with 14 parts.
Licence	open source
Operating System	windows/MacOS/Linux
URL	`http://cs.gmu.edu/~eclab/projects/mason`

Table B.5: Overview Summary of MASON.

Version	0.4.1 compared (current 0.6)
What is MATSim?	MATSim is a Java-based Multi-agent simulation toolkit with the focus of transport.
Developers	MATSim is a joint project of Transport Systems Planning and Transport Telematics, Technical University of Berlin and the Institute for Transport Planning and Systems (IVT), Swiss Federal Institute of Technology Zurich.
Properties	is designed to simulate whole days, was specifically made for traffic scenarios but can also be used for evacuation scenarios, supports modeling for public transport.
Features	offers multicore support, developer do yearly meetings with MATSim users, there are several extensions on the website.
Further reading	a list of the most important publications is on the homepage of `matsim.org/publications` which lists from 2003 until 2014.
Documentation	MATSim offers several tutorials for beginners and a developer guide for advanced researchers. Additionally there are FAQ and mailing lists to answer questions.
Licence	open source
Operating System	windows/MacOS/Linux
URL	matsim.org

Table B.6: Overview Summary of MATSim.

graphical options of the simulation are very limited. An integrated development environment for better representation of MASON would have been useful.

B.5.4 Conclusion

Without the GeoMason extension offers MASON only little support to create traffic simulation. The requirement of communication is satisfied. With lots of effort an comprehensive traffic simulation could be build from scratch. With detailed examination the graph-based approach does not seem ideal for traffic scenarios. Other simulation platforms offer better solutions to create an appealing simulation environment. For advanced Java programmers is MASON good if they focus on the performance of the platform and not the environment.

B.6 MATSim: Multi-Agent Transport Simulation Toolkit

Background information about MATSim is already provided in **??**. Here more detailed and summarized information are given.

B.6.1 Installation

MATSim uses the current Java Runtime Environment (JRE). The developer proposes the use of Eclipse Classic as an developing environment. In order to use the functionalites of MATSim the zip file in the download area from sourceforge.org. Then the zip file needs to be integrated into the Eclipse Project. The visualizer via (`senozon.com/matsim/via/download`) is needed for visualizing the calculated simulations graphically and it needs to be installed separately. Note that the free license file needs to be asked for via email for using the visualization software. It also uses Maven and nightly bites if required for developers. The current user guide does not work - last update from Fri, 2008-07-18 23:15 matsim.

A useful extension is the networkEditor from TU Berlin (`matsim.org/extensions/networkEditor`) where road networks can be modified from data of OpenStreetMap. The networkEditor also allows to export shape files which are used for MATSim.

Another extension is MATSIM4UrbanSim (`matsim.org/extensions/MATSIM4UrbanSim`) which connects MATSIM with the Simulation system UrbanSim (`urbansim.org`) where 3D environments can be modeled and scenarios be used for city planing.

B.6.2 Advantages

A user with little knowledge can uses MATSim for an individual traffic simulation based on real maps. MATSim has a routing algorithm and car following model implemented which is activated when agents are executed. Metadata for map material like speed limits or traffic directions are automatically transfered. The separation of the calculation and graphical output of that data with the visualizer is good for big and complex simulation. For individual focus of agents MATSim offers display options of the plans. Also buildings and the possibility of opening times extends the simulation environment.

B.6.3 Disadvantages

The before mentioned separation of the calculation and visualization likewise means that influencing the running simulation from the visualizer is not possible. The support of the analysis of the data is not extensive and the contact to the developers is a bit long-winded through mailing list. But the biggest critics is that there is the lack of Vehicle-to-X communication. MATSim is designed to simulate complete cities and countries instead of small area focuses for vehicle group behavior.

B.6.4 Conclusion

MATSim is a specialized simulation software for the use of traffic and evacuation scenarios. The creation of a traffic simulation based on real maps is easy and good. But the fact that MATSim does not support communication, makes it impossible for cooperative traffic scenarios. During selection of the agent-oriented simulation platforms it was not obvious if this communication feature is supported or not. However, MATSim is a well functioning and good documented active simulation platform. But for the thesis purpose it is not suitable.

B.7 NetLogo

B.7.1 Installation

The download area of the homepage NetLogo provides a setup-file. After installation NetLogo is started with the .exe file and no other actions are necessary to use NetLogo.

B.7.2 Advantages

NetLogo offers a big sample modeling library for different domains. The tutorial is easy to understand and useful. Using NetLogo is intuitive and uncomplicated. It is possible to make changes through a slide controller during the running simulation. With the applet export function simulation models can be displayed in the browser.

B.7.3 Disadvantages

In NetLogo projects the code of the simulation is located in the code-rider. Because all program lines are written in this rider, the code gets unclear and confusing in bigger projects. Therefore the modeling of complex projects is problematic. Another problem is that NetLogo is grid-based, but continous space is better for realistic traffic simulation.

B.7.4 Conclusion

NetLogo is an all-purpose candidate for the creation of small agent-oriented simulation. The handling is uncomplicated and very good for first-time users. The sample scenarios for the traffic cause that NetLogo has a better ranking than the alternatives SeSAm or Repast S, although the functions of the platforms are quite similar.

B.8 Repast S

B.8.1 Installation

Repast S provides a setup file for download. The installation contains an adapted Eclipse version as the developer environment with some example models and tutorial in format of .pdf. Repast S offers the tutorial in the programming languages Relogo and Java, but not different in topic.

Version	5.0.1 compared (current 5.2)
What is NetLogo?	NetLogo is a all-purpose programmable modeling environment for multi-agent simulation.
Developers	Originally NetLogo was developed by Uri Wilensky in the year 1999. Since then it is continuously developed by the Center for Connected Learning (CCL) and Computer-Based Modeling close to Chicago, Illinois, USA.
Properties	NetLogo supports a simple but still powerful programming language based on Logo. It supports 2D and 3D models and has a grid-based simulation environment. NetLogo provides models for system dynamics.
Features	NetLogo possesses a very big sample application library from many different domains. With the extension HubNet simulation can be distributed on different computer connected in a network. NetLogo allows to change the simulation model during a running simulation.
Further information	NetLogo was developed for the education domain. An overview of the publication is listed on the homepage of the CCL. There is book "Agent-Based and Individual-Based Modeling: A Practical Introduction" [282] which uses NetLogo as an simulation environment.
Documentation	NetLogo provides a tutorial in 3 parts which are very good for first-time user. There are useful guides to explain the functions of NetLogo. Also a FAQ area is on the website.
Licence	open source
Operating System	windows/MacOS/other operating system with Java 5 installed
URL	`http://ccl.northwestern.edu/netlogo`

Table B.7: Overview Summary of NetLogo.

Version	2.0 compared (current 2.3.1)
What is Repast?	Repast Simphony or short Repast S is a multi-agent simulation toolkit with the focus on social science. Repast S can be programmed in Java or a programming language called Relogo.
Developers	Repast was developed in the department of Decision and Information Sciences in the Argonne National Laboratory in Lemont, Illinois, USA.
Properties	Repast is a tool for graphic model development in form of flow charts, support of 2D and 3D models, grid-based simulation environment, automatized database connection and reporting function in form of logs.
Features	Repast S supports lots of external tools to analyze the simulation scenarios like spreadsheets, with the extension HLD simulation can be distributed in a network of connected computer, possibility to model agent behavior in form of flowcharts.
Further information	Repast was originally a Java version of the simulation platform Swarm. The last Version Repast-3 was then detached with the further development of Repast Simphony.
Documentation	Repast S offers starter tutorials with the programming language Java and an instruction to create flowcharts. An API documention and FAQ are also available.
Licence	open source
Operating System	windows/MacOS/Linux
URL	`http://repast.sourceforge.net`

Table B.8: Overview Summary of Repast Simphony.

B.8.2 Advantages

The advantage of Repast S is that simulation can be developed with different programming languages. Repast S offers a big selection of export possibilities which are easy to use. The requirement of communication support is satisfied. Modeling with flowcharts makes the use easy. It is designed for Social Science. Just like NetLogo it is a good platform for first-time users.

B.8.3 Disadvantages

The tutorial is quite short. There are no sample simulation from the traffic domain available. The handling of errors is strange, patches for the modeling of complex environments are not adequate. It is grid-based like NetLogo.

B.8.4 Conclusion

The framework of Repast S and NetLogo are very similar. Both simulation platforms use turtles as agent containers, as well as patches for the modeling of the simulation environment as core elements of the simulation. This is better for simple than for complex scenarios. Repast S supports different programming languages and the selection of export possibilities. In direct comparison NetLogo seems to be more mature.

B.9 SeSAm: Shell for Simulated Agent Systems

B.9.1 Installation

SeSAm can be installed with a setup file which is available on the website. For Windows it is an .exe file and after installation no more steps are needed for using SeSAm.

B.9.2 Advantages

The installation was very easy and only little programming knowledge is necessary to create a simulation with SeSAm. The lists of simulation components is well structured and the summarized presentation of the agent behavior in form of an UML-similar activity diagram is clearly represented. The requirement of communication is satisfied.

B.9.3 Disadvantages

Difficulties in usability: due to the nesting of function the code is hard to read. It is complicated to find out what a new created function causes as results. It is also difficult to find the reason of errors. Due to copy and paste as well as do and undo operations, SeSAm can have problem with the usage of the code compared to text-based developer environments. The use of resource objects for the representation of street segments is not really appropriate.

Version	2.5.1 compared (current 2.5.2)
What is SeSAM?	SeSAm offers a general developing environment for multi-agent simulation.
Developers	SeSAm was developed on the Chair of Artificial Intelligence and Applied Informatics of the University of Würzburg., Germany.
Properties	SeSAm possesses a graphic user interface and has integrated analyzing functions to evaluate the simulation data, SeSAm offers the possibility to save several scenarios in one project.
Features	SeSAm offers a FIPA Plugin for agent communication which makes it compatible to JADE (described in Subsection B.3), the execution of a simulation can be distributed within a network.
Further reading	In the SeSAm paper [205] the authors describe the components of SeSAm briefly and the special features of the simulation environment for visualizing agent behavior.
Documentation	There is a large wiki and comprehensive tutorial and also FAQ.
Licence	open source
Operating System	windows/MacOS/Linux
URL	`http://www.simsesam.de`

Table B.9: Overview Summary of SeSAm.

B.9.4 Conclusion

SeSAm offers a good all-round package: clean user interface, support of communication and plugins which extend the functionalities of SeSAm. The problems in usability and the limited possibilites of modeling the simulation environment show that SeSAm is not ideal for the purpose of this work.

B.10 Summary Agent Simulation for Cooperative Traffic

In the evaluation of the master thesis by `Eichhorn` [97] the general overview of the agent-oriented simulation for cooperative traffic, the best total results are for AnyLogic (after 30 days test, commercial software) and NetLogo. In the ranking the platforms Repast S, Jason, SeSAm, and MASON are also appropriate for the purpose. Whereas MATSim (rank 7) and JADE (rank 8 - as standalone, better in combination as a middleware) are not very useful for agents in cooperative traffic with respect of the categories environment, interaction, algorithms and individualization.

For the choice of a simulation platform, it is important the way the traffic scenario is created. For scenarios with not much detail and based on real map data, MATSim and MASON with the Geomason extension are useful. For scenarios where communication and the modeling of complex agent behavior are preferred, then Jason and JADE are good options. The best all-in-one packets for support of modeling the environment, possibilities to program agent-oriented behavior and easy-to-learn offer AnyLogic, NetLogo, Repast S, and SeSAm.

No grid-based simulation environment seems appropriate like Repast S and NetLogo, but continuous space seems more functional - since traffic is fluent and the later is more realistic. BDI agent model seems to be state-of-the-art and should be preferred when choosing the agent platform which is provided by Jason. The FIPA communication standard is supported by Jason and JADE as a middleware.

For grouping vehicles especially individualization and interaction are needed and the environment and algorithms could also be done with tools from the traffic simulation for more mature results. Therefore, the preferences for this thesis are Jason or JADE as agent interpreter, whereas for very realistic environments with integrated algorithms, there are more details needed than the more macroscopic street map data of MATSim or MASON are offering.

Men are only as good as their technical development allows them to be.
George Orwell (1903-1950)

Appendix C

Technical Aspects of MATI

The MATI approach modifies, uses and extends the AplTk and EIS developed in the PhD thesis by `Behrens` [31] described in the Background Section 2.3.6 to connect various agent interpreters with different traffic simulators (such as AIMSUN, MovSim[1], MASSim[2] or SUMO) as shown in Figure 4.13 with the gray wild-card characters. Instead of writing a new framework from scratch (like Streetworld in Section 4.3.1) MATI uses existing and well-established modules (good for non-functional requirements). The challenge of combining a simulation environment with an interpreter (such as Jason[3] or JADE[4] like done in `Görmer et al.` [138]) provided additional motivation to develop the MATI architecture. The claim is that MATI simplifies the integration of JAVA-based MAS which support Belief-Desire-Intention (BDI) architectures with discrete (traffic) simulators.

The yellow package in Figure 4.13 describes the Agent Programming Language Toolkit Extended (AplTkEx) as an extention of AplTk. AplTkEx is a framework where the agent side is implemented with the interpreter interface and the environment side is directly connected to the existing EIS-Interface colored in green. Technically, AplTkEx extends the EIS with a step()-method. In every simulation core step the analyzing tools evaluate the interpreter and the environment steps and send the result to AplTkEx. Lower-level MATI packages appearing in red and combined for MATI connect directly to the tool, environment and interpreter interfaces of AplTkEx.

The environment and interpreters connect to the *Environment Interface Standard* in the EIS package and is used as described in Section 2.3.6.

The addition of the package AplTkEx creates the possibility that environments and interpreters can execute multiple steps in a single core step. This allows the operating speed of the agent to be manipulated in proportion to the

[1] *MovSim* available at `http://movsim.org`
[2] *MASSim* available at `http://multiagentcontest.org`
[3] *Jason* available at `http://jason.sourceforge.net`
[4] *JADE* available at `http://jade.tilab.com`

simulation time. After the interpreter, environment and tool components, the step results will be evaluated in every core step. Thus, in AplTkEx the tools can additionally evaluate environment steps, the core states are modifiable and the control functions are improved compared to the used AplTk by `Behrens` [31].

AplTkEx consists of four main parts: interpreters, environments, tools (e.g. visualization) and the core. Figure C.1 shows the AplTkEx structure with modifications to the AplTk infrastructure (c.f. [31] p. 131). In AplTkEx tools do not control the core. The core manages the work processes of the components and optional environments. Tools are added separately. In principle, this allows environments other than traffic simulators to be supported. While AplTkEx provides the interfaces for the three components interpreter, environment and tool, the specific bindings need to be implemented. These bindings are defined in the process description and are loaded as a jar-file during runtime. An instance of AplTkEx consists of at least one core and one interpreter. That means one agent simulation can run on its own and is managed by the core.

The **core** is the central component of the AplTkEx and serves as the controller. For the simulation the core initializes all necessary components described in the process description and runs the simulation with the AplTkEx-Run method. The AplTkEx-Run of Core-Run is shown in Algorithm C.

```
ForEach{scenario of scenarios}{\\
   For{i\leftarrow 0 To scenario.repeats+1}{
  instantiate tools\;
  instantiate interpreters\;
  instantiate environments\;
  For{$i\leftarrow 1$ To $scenario.steps$}{
      environments execute step\;
    interpreters execute step\;
    tools process environment steps\;
    tools process interpreter steps\;
  }
  release tools\;
  release interpreters\;
  release environments\;
  }
}
```

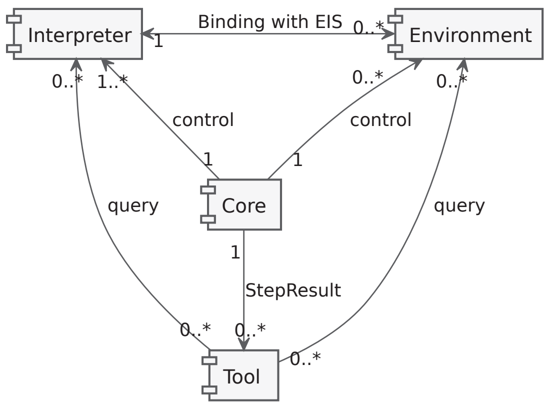


Figure C.1: AplTkEx Structure with Modifications to AplTk (c.f. [31] p. 131): Tools do not control the core; actions and percepts are realized through the EIS binding.

AplTkEx-Run is executed sequentially. Therefore all scenarios which are defined in the process description will run in the composed order. Every scenario will be executed at least once, but additional execution repetitions can be defined. In a Core-Run environments and interpreters will be initialized according to the defined process description. The *interpreter* is interlinked to the specified *environment* where its entities will connect to the agents of the interpreter. Every interpreter can be coupled to one environment whereas one environment can be attached to multiple interpreters. Subsequently the predefined tools are initialized and then the execution phase starts. When the declared scenario.steps in the algorithm C are fulfilled, the environments, interpreter and tools execute sequentially one step. After the execution phase all components will quit.

AplTkEx has several states of operation like seen in Figure C.1; MATI extends the AplTk.Core with 'INIT', 'INIT_SCENARIO', 'SCENARIO_FINISHED', 'ABORTING' and 'ABORTED'. Figure C.2 illustrates the states and transactions of Core::State. The grey states derive from the AplTk states 'PAUSED', 'RUNNING' and 'FINISHED'; states that are newly added in AplTkEx are denoted in white. The states regulate which methods have access, e.g., to determine when the start and pause function of the AplTk.AppWindow are valid.

All states and their relevance are described here:

INIT is the initial state of the core.

INIT_SCENARIO state initializes all scenario-specific components.

PAUSED in this state the simulation is at rest. When the scenario initialization is complete, the core is transfered automatically into PAUSED. This state can also be reached when the user interrupts the AplTkEx-Run.

RUNNING is the opposite to PAUSED and is executed in AplTkEx-Run.

FINISHED_SCENARIO This state is reached when the scenario is complete. It can be repeated, the next scenario can be loaded or the AplTkEx-Run can be finished.

FINISHED is the state when all scenarios are executed.

ABORTING the user can initiate this state to stop the AplTkEx-Run.

ABORTED this state is reached when the abortion is successfully performed.

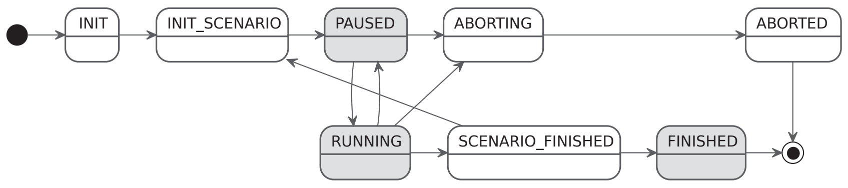


Figure C.2: AplTkEx Core States. White Core States represent extensions to AplTk.

The **tool** serves in the AplTkEx as the analyzing and evaluation instrument. Tools needs to implement the interface ToolIF. Within every cycle of AplTkEx-Run tools can request and evaluate the step-results of specific interpreters and environments.

The **environment** represents the world context in which the entities are situated. Agents of the interpreter which are interlinked with the environment via EIS can read and control those entities. Environments implement the EnvironmentIF (as illustrated in Figure 4.13) and one environment can be connected to diverse interpreters. Thus, it is possible for agents from different platforms like Jason or pure JAVA to interact in the same world. The **interpreter** represents the multi-agent simulation of AplTkEx and administrates agent interactions in the simulation. The interpreter needs to implement the interface InterpreterIF. In every cycle of AplTkEx-Run the InterpreterIF::step() is called. In this step operations like agent cycles can be executed. The interpreter can be connected to an environment with EIS. The environment entities are controllable through the agents. More than one environment is technically possible, but does not seem reasonable due to conflicting perceptions from different environments.

The environment is created with the microscopic traffic simulation tool SUMO for the decision logic of Traffic Management Center (TMC) and basic (default) vehicle behavior. Special attention is given to organizational aspects the decentralized grouping decisions of the vehicles, supported by corresponding communication. This behavior is implemented in the agent interpreter Jason.

C.1 Installation of MATI

At present the environment SUMO, the interpreter Jason, and the tool HDF5 are used and need to be installed in the following versions:

- JDK 7 or higher [5]
- Maven 2.3 or higher [6]
- HDF5 [7] and HDF-JAVA 2.10 [8]
- SUMO 0.20.0 or higher [9]
- up to date driver for graphics with openGL is mandatory! especially for Windows!

C.1.1 SUMO

SUMO is downloaded from its website [10] and depending on the used operating systems installed (compare SUMO wiki[11]). The bin-directory of SUMO needs to be inserted within the path-environment variable in order to employ an automatized call. For the scenario configuration and structure there is an instruction in target/src/scenarios.

[5] `http://www.oracle.com/technetwork/java/javase/downloads/`
[6] `http://maven.apache.org/`
[7] `http://www.hdfgroup.org/HDF5/release/obtain5.html`
[8] `http://www.hdfgroup.org/products/java/hdf-object/`
[9] `http://sumo-sim.org/`
[10] `http://www.sumo.dlr.de/wiki/Downloads`
[11] `http://sumo.dlr.de/wiki/Main_Page`

SUMO is provided with an appropriate installation and is located for all operating systems in 'lib/sumo'. For the scenario configuration and structure is located in 'target/src/scenarios'.[12] The bin-directory of SUMO needs to be set within the path environment variables in order to allow automatized call. For the scenario configuration and structure is a manual under target/src/scenarios.

The following items need to be respected to execute MATI including SUMO:

MATI needs to be built and installed with Maven (mvn) install and start with runmati.bash (Linux / OSX) or runmati.bat (Windows) for the scenario call

```
'runmati.bat / runmati.bash <target/scenarios/
<Scenariodirectory>/scenario.json'
```

The following definitions apply for the scenario configuration in MATI:

- SUMO Executables (SUMO or SUMO-GUI) need to be accessible within the path environment variables
- relative path details use Unix notation
- directory and file names use lower case (also within the configuration use correct notation!)
- use structure of directories as described in the next paragraph
- MATI Jars are situated on the same level like "Scenario" file, such that the path specification need to be relative to it.
- one directory contains the exact files which are necessary for the scenario. Thus, one scenario = one directory.
- not required or temporary files need to be eliminated out of the directory
- within the "common" folder are the files which are used for all scenarios (global)

C.1.2 Maven

The following items need to be respected for executing MATI including SUMO:

- MATI needs to be build and installed with 'mvn install'
- If SUMO shall be started with GUI, then the environment variable 'MATI_GUI' needs to be set with an not empty value. Otherwise if the variable is not set, MATI and SUMO are executed without GUI visualization.
- MATI is started in the following manner:
  1. if necessary built MATI
  2. set environment variable for the GUI

[12]Further instructions: (`https://mecdev.rz-housing.tu-clausthal.de/gitlab/matigroup/mati/tree/master/src/main/resources/scenarios`)

3. using the scenario with the amount of runs is called

```
runmati.bash <target/scenarios/<scenariodirectory> <executionnumber> (Linux / OSX) or
    runmati.bat  <target/scenarios/<scenariodirectory> <executionnumber> (Windows)

```

Website: `http://maven.apache.org/`

If dependencies are existent in online MavenRepos, then add them. Dependencies can be found on `http://mvnrepository.com/`, then add the corresponding pom.xml

C.1.3 Dependencies

```
<dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.0</version>
      <scope>test</scope>
</dependency>
```

is inserted in

```
<dependencies>
      ...
</dependencies>
```

else load dependency via local Repository (Repo)

- create local Repo
- create directory "lib" in Maven-Project-Directory and
- insert the following code into pom.xml

```
<repositories>
    <repository>
        <id>lib</id>
        <name>lib</name>
        <releases>
            <enabled>true</enabled>
            <checksumPolicy>ignore</checksumPolicy>
        </releases>
        <snapshots>
                <enabled>false</enabled>
        </snapshots>
        <url>file://${project.basedir}/lib</url>
    </repository>
</repositories>
```

add dependencies to local Repo and execute following instructions

```
cd [MavenProjectDir]
mvn install:install-file -Dfile=[myArtifact.jar] -DgroupId=[x.y.z] -DartifactId=[artifactId] \
-Dversion=[version] -Dpackaging=jar -DgeneratePom=true
mkdir -p [MavenProjectDir]/lib/[X]/[Y]/[Z]/
mv ~/.m2/repository/[X]/[Y]/[Z]/[artifactId] [MavenProjectDir]/lib/[X]/[Y]/[Z]/[artifactId]
```

write dependency definition into pom.xml like described above

C.1.4 Example

```
cd ~/git/mati/core
mvn install:install-file -Dfile=jason.jar -DgroupId=net.sourceforge.jason -DartifactId=jason \
-Dversion=1.3.9 -Dpackaging=jar -DgeneratePom=true
mkdir -p ~/git/mati/core/lib/net/sourceforge/jason/
mv ~/.m2/repository/net/sourceforge/jason/jason ~/git/mati/core/lib/net/sourceforge/jason/jason
```

```
<dependency>
    <groupId>net.sourceforge.jason</groupId>
    <artifactId>jason</artifactId>
    <version>1.3.9</version>
    <scope>compile</scope>
</dependency>
```

C.1.5 Instructions

- mvn versions:display-plugin-updates
- mvn versions:display-dependency-updates

C.1.6 HDF5

The HDF library is provided for all operating systems and is located in 'lib/hdf'. For example:

```
  runmati.bash target/scenarios/jana/homogen/three_lanes 10
  runmati.bat  target\scenarios\jana\homogen\three_lanes 10
```

The following items are for the execution of the MATLAB analysis of the hdf files important: In order to use dimensionality reduction functions install the drtools: `http://lvdmaaten.github.io/drtoolbox/`

Under this web page you can find the download link. Unzip the downloaded folder into your toolbox directory under your matlab installation. You can further read the installation instructions in the readme file provided from the drtoolbox directory.

MATLAB needs to be installed and registered with the PATH. Additionally to the MATLAB installation add in the line of ' /.bashrc' the following at the end of the file:

```
  export PATH=/path/to/matlab/bin:$PATH
    export PATH=/path/to/matlab/:$PATH
```

The MATLAB evaluation is opened and started under Linux with

```
  matlab -r "mati_evaluation('<h5_file_name.h5>')"
```

after the execution of the MATLAB script every scenario has created two avi-files in the MATI file. One for the 'distables' and another for the 'paramstables' the dimensions of the matrices come from the tool-files under

```
   mati/src/main/resources/scenarios/jana/common/../toolFiles/tool.json
```

and from the 14 predefined characteristics, which are defined in

```
mati/groupingbenchmarktool/src/main/java/mati/tools/groupingbenchmarktool/CGroupingBenchmarkTool.java
```

.

C.2 Directory and file structure

```
ScenarioName
   |
   |---environmentFiles
   |         |
   |         |----- scenario.sumocfg (SUMO configuration)
   |         |----- net.xml (SUMO netfile)
   |         |----- route.xml (SUMO routefile)
   |         |
   |         |----- polygon.xml (optional SUMO polygonfile)
   |         |
   |         |
   |         |----- build (optional file for creating SUMO data)
   |                  |
   |                  |----- build.xml (SUMO Net-Converterfile)
   |                  |----- edge.xml (SUMO Edgefile)
   |                  |----- node.xml (SUMO Nodefile)
   |                  |----- typ.xml (SUMO Typefile)
   |                  |----- connection.xml (SUMO Connectionfile)
   |
   |
   |
   |---interpreterFiles
   |         |
   |         |----- interpreter.json (MATI Interpreter File)
   |         |
   |         |----- *.asl Dateien (Jason Agent-Speak Files)
   |
   |
   |
   |---toolFiles
   |         |
   |         |----- tool.json
   |
   |
   |
   |--- scenario.json (main configuration of the scenario for starting MATI)
```

C.2.1 Scenario configuration

The software SUMO can be primary used or extended.

For only using SUMO: download, install the binaries and run the pre-built packages. Everything is already included, which is needed to run SUMO. It is recommended to install Python for executing the additional scripts in the tools-folder.

For extending SUMO (the environment is extended with interpreter in MATI): download the source distribution depending on the used operating system (Windows, Linux, or MacOS) and build SUMO on the system of choice with the pre-built binary distribution.

The following instructions were called on Ubuntu 14.04 as one possible configuration.

- download sumo-src-0.21.0.tar.gz from "http://sumo-sim.org/userdoc/Downloads.html" and unpack it in a desired folder (right-click on the downloaded file and select 'Extract Here')
- download xerces-c-3.1.1.tar.gz from "http://xerces.apache.org/xerces-c/download.cgi" and unpack it in a desired folder (see above)
- go to the xerces folder which is just unpacked and run the following commands:
  - >> ./configure
  - >> make
  - >> sudo make install
- now go to the unpacked SUMO folder and run the following commands:
  - >> ./configure –with-xerces-libraries='path to the unpacked xerces folder/xerces-c-3.1.1/src/.libs'
  - >> make
  - >> sudo make install
- go to the bin directory under the xerces folder and run the SUMO-GUI as shown below:
  - /xerces-c-3.1.1/bin$ ./sumo-gui
  - or
  - /xerces-c-3.1.1/bin$ ./sumo
- run SUMO from every destination. For example, in the home folder, just type the following:
  - >> sumo-gui

OSX (10.9) Dependencies: `http://xquartz.macosforge.org/landing/` `http://brew.sh/` (recommended) or `http://www.macports.org/`

```
brew install fox gdal proj xerces-c autoconf
     pkg-config libtool automake
```

```
svn co \url{http://svn.code.sf.net/p/sumo/code/trunk/sumo sumo-trunk}
cd sumo-trunk
export CPPFLAGS="$CPPFLAGS -I/usr/local/include -I/usr/X11R6/include"
export LDFLAGS="$LDFLAGS -L/usr/local/lib -L/usr/X11R6/lib"
make -f Makefile.cvs
./configure --with-fox-config=/usr/local/bin/fox-config
  --with-proj-gdal=/usr/local --with-xerces=/usr/local --prefix=/usr/local
make
```

Configuration SUMO:

```
set SUMO_HOME with the Path to the sumo-dir
modify $SUMO_HOME/data/xsd/routeTypes.xsd:
    add <xsd:attribute name="agenttype"
    type="xsd:string"/> into the parent
    <xsd:complexType name="vehicleType">
```

Car-Following Models in SUMO:

- carFollowing-Krauss (SUMOKrau): The Krau-model with some modifications which is the default model used in SUMO
- carFollowing-KraussOrig1 (SKOrig): The original Krau-model
- carFollowing-PWagner2009 (PW2009): A model by Peter Wagner, using Todosiev's action points
- carFollowing-BKerner (Kerner): A model by Boris Kerner (Caution: currently under work)
- carFollowing-IDM: The Intelligent Driver Model (IDM) by Martin Treiber (Caution: Problems with lane changing occur)

Interesting

- with ”Abstract Vehicle Class” exclusive lanes can be defined
- with ”guiShape” different vehicle types can be added to predefined shapes
- random ”vTypes”- and ”route”-distribution via ”vTypeDistribution” and ”routeDistribution”
- the stopping of vehicles can be defined as ”stop” child of ”vehicle” and thus, ”stop” is also a TraCI variable of ”Change Vehicle State”

Links
`http://sumo-sim.org/userdoc/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html` `/mati-data/docs/SUMO_Cookbook.pdf`
To create a SUMO scenario the following files are needed:

1. Name.nod.xml
2. Name.edg.xml
3. Name.typ.xml (optional)
4. Name.con.xml (optional)
5. Name.rou.xml
6. Name.net.xml
7. Name.sumocfg.xml

Name must be the same in all data of your scenario. Hint for MATI:

- change of vehicle types is made in janaScen.rou.xml
- changes of edges are made in janaScen.edg.xml

In summary, the designer needs to proceed as follows (here for the Hildesheimer net):

1. generate net.xml -file: **netconvert -n build/node.xml -e build/edge.xml -v -t build/type.xml**
2. then create od2trips: **–od-matrix -files=hildesheimer_od_matrix.txt –net-file=hildesheimer.districts.xml –output-file=hildesheimer.trips.xml –timeline.day-in-hours-true**
3. last use the duarouter: **-n hildesheimer.net.xml -t hildesheimer.trips.xml -o hildesheimer.test.rou.xml**

Creating a Net

Net First, a *nod.xml*, a *edg.xml*, a *typ.xml*, and a *con.xml* need to be created like shown in the listings above. The creation of these files is described on `http://sumo-sim.org/userdoc/Tutorials/Quick_Start.html`. The created file describing the new net in SUMO (net.xml) by defining a netc.cfg file where input and output files are defined used by *netconvert*. The creation of the netc.cfg file is described in the above stated url. With *netconvert* the new net.xml file is produced:

```
>> netconvert -c <your\_file\_name>.netccfg
 (the '>>' is indicating the command line).
```

Nodes In the file Name.nod.xml all nodes of the network are defined.
Example:

```
<node id=  n o d e X   x=  0 .0   y=  1 5 0   />
```

Every node has a unique id and a position within the SUMO world. Consider the numbers x and y in pixels. The position (0/0) is not automatically in the top left corner of the simulation window. It depends on the other nodes as well. Nodes are the starting and ending point in x and y coordinates of a street, thus, as well for a lane or segment. The x and y coordinates as a node get and id which makes it easier to find and identify them. On the SUMO examples of the tutorial [225], the nodes are created as follows in the xml scheme:

```
<?xml version="1.0"?>
-<nodes xsi:noNamespaceSchemaLocation=
"http://sumo.dlr.de/xsd/nodes_file.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<node y="0" x="0" id="dl"/>
<node y="0" x="1200" id="dr"/>
<node y="190.99" x="0" id="tl"/>
<node y="190.99" x="1200" id="tr"/>
</nodes>

********* here starts the three lanes homogen scenario *********
<?xml version="1.0" encoding="UTF-8"?>
-<nodes xsi:noNamespaceSchemaLocation=
"http://sumo.sf.net/xsd/nodes_file.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<node y="0.0" x="0.0"  id="startpre" type="traffic_light"/>
<node y="0.0" x="1000" id="startmain type="traffic_light""/>
<node y="0.0" x="2000" id="startpost" type="traffic_light"/>
<node y="0.0" x="3000" id="sink" type="traffic_light"/>
</nodes>
```

Listing C.1: Description of Nodes.

The 'three lanes' scenario has a divided length of three equal segments: from source to startmain for the pre-phase, then the main phase and a post phase which ends into the sink. This is also the id names. Additionally the type 'traffic light' is defined.

Edges This file contains all the edges of the network to be created.
Example:

```
<edge id=   x 1    from=   n o d e X    to=   n o d e Y    type=   t 1    />
```

If the type attribute is used, also the Name.typ.xml-file is needed where more attributes for edges like speed or number of lanes are defined. The attributes can be added to the edge directly as well, but it is recommended to use the type file for more organization in code. nodeX and nodeY are node-ids of defined nodes in the Name.nod.xml-file. Speed is measured in m/s.

```
<?xml version="1.0" encoding="UTF-8"?>
-<edges xsi:noNamespaceSchemaLocation=
"http://sumo.dlr.de/xsd/edges_file.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<edge to="2" id="1to2" from="1"/>
<edge to="3" id="out" from="2"/>
</edges>
************** here three lane scenario ************************
<?xml version="1.0" encoding="UTF-8"?>
-<edges xsi:noNamespaceSchemaLocation=
"http://sumo.sf.net/xsd/edges_file.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<edge type="a" to="startmain" from="startpre"  id="prephase"/>
<edge type="a" to="startpost" from="startmain" id="mainphase"/>
<edge type="a" to="sink" from="startpost" id="postphase"/>
</edges>
```

Listing C.2: Description of Edges.

Lane Types Here types of edges also known as lanes are defined.
Example:

```
<type id=   t 1    priority=   3    numLanes=   3    speed=   1 5   />
```

```
<?xml version="1.0" encoding="UTF-8"?>
-<types>
<type speed="13.889" numLanes="3" priority="3" id="a"/>
<type speed="13.889" numLanes="2" priority="3" id="b"/>
<type speed="13.889" numLanes="3" priority="2" id="c"/>
</types>
*********** three lane scenario ***************************
<?xml version="1.0" encoding="UTF-8"?>
-<types>
<type speed="10" numLanes="3" priority="3" id="a"/>
</types>
```

Listing C.3: Description of Lane Types.

Connections Connections are not mandatory, but can represent the turnings and need to be defined.

```
<?xml version="1.0" encoding="UTF-8"?>

-<connections xsi:noNamespaceSchemaLocation=
"http://sumo.dlr.de/xsd/connections_file.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<connection toLane="0" fromLane="0" to="L12" from="L2"/>
<connection toLane="1" fromLane="0" to="L12" from="L2"/>
<connection toLane="2" fromLane="1" to="L12" from="L2"/>
<connection toLane="0" fromLane="0" to="L14" from="L4"/>
<connection toLane="1" fromLane="1" to="L14" from="L4"/>
<connection toLane="2" fromLane="1" to="L14" from="L4"/>
<connection toLane="0" fromLane="0" to="L11" from="L9"/>
<connection toLane="1" fromLane="1" to="L11" from="L9"/>
<connection toLane="2" fromLane="1" to="L11" from="L9"/>
<connection toLane="1" fromLane="1" to="L15" from="L9"/>
<connection toLane="2" fromLane="2" to="L15" from="L9"/>
<connection toLane="0" fromLane="0" to="L10" from="L16"/>
<connection toLane="1" fromLane="1" to="L10" from="L16"/>
<connection toLane="2" fromLane="1" to="L10" from="L16"/>
<connection toLane="2" fromLane="2" to="L11" from="L16"/>
<connection toLane="0" fromLane="0" to="L15" from="L12"/>
<connection toLane="1" fromLane="1" to="L15" from="L12"/>
<connection toLane="0" fromLane="1" to="L10" from="L12"/>
<connection toLane="1" fromLane="1" to="L10" from="L12"/>
<connection toLane="2" fromLane="2" to="L10" from="L12"/>
<connection toLane="1" fromLane="1" to="L16" from="L14"/>
<connection toLane="0" fromLane="1" to="L16" from="L14"/>
<connection toLane="2" fromLane="2" to="L16" from="L14"/>
<connection toLane="0" fromLane="0" to="L18" from="L14"/>
<connection toLane="1" fromLane="1" to="L18" from="L14"/>
<connection toLane="2" fromLane="1" to="L18" from="L14"/>
<connection toLane="0" fromLane="0" to="L16" from="L17"/>
<connection toLane="1" fromLane="1" to="L16" from="L17"/>
<connection toLane="2" fromLane="1" to="L16" from="L17"/>
<connection toLane="0" fromLane="1" to="L13" from="L17"/>
<connection toLane="1" fromLane="1" to="L13" from="L17"/>
<connection toLane="2" fromLane="2" to="L13" from="L17"/>
</connections>
******************three lane scenario *******************
**** no need of turns, therefore no connections *********
```

Listing C.4: Description of Connections.

Filling the Net

OD-Matrix In order to specify the amount of vehicles in the new scenario and to set the routes of these vehicles, an OD-Matrix can be defined as follows:

```
$VMR
* vehicle type 4
* From-Time 0.00 To-Time 1.00
* Factor 0.05
*
* some additional comments
* District number 10
* names: hildesheimerSouthInwards altenWest altenEast geibelWest
geibelEast krausenWest krausenEast aegiWest aegiNorth schlaegerEast
*
* District hildesheimerSouthInwards Sum = 11451
  0 454 914 0 46 60 364 4720 4893 0
* District altenWest Sum = 3078
  860 0 2211 0 7 0 0 0 0 0
* District altenEast Sum = 8109
  301 3108 0 295 920 1 125 240 3119 0
* District geibelWest Sum = 2263
  0 0 462 0 60 0 609 9 1123 0
* District geibelEast Sum = 682
  69 15 3 71 0 0 84 69 371 0
* District krausenWest Sum = 1370
  329 0 2 0 0 0 330 0 654 0
* District krausenEast Sum = 2297
  350 139 10 767 0 559 0 468 4 0
* District aegiWest Sum = 19153
  4042 0 247 0 100 0 0 0 13991 773
* District aegiNorth Sum = 6831
  3773 22 13 531 1 1552 663 0 0 276
* District schlaegerEast Sum = 738
  0 0 0 0 0 0 0 38 700 0
```

Listing C.5: Description of an OD-matrix.txt file.

Districts In order to create an OD-matrix districts inside a districts.xml file need to be specified. Districts are defined by a source and a sink. Sources and sinks are edges in a traffic network, which were previously defined in the edg.xml file. Districts are created as follows:

```
<tazs>
  <taz id="hildesheimerSouthInwards">
      <tazSource id="altensouth-alten" weight="1" />
      <tazSink id="alten-altensouth.252" weight="1" />
    </taz>
</tazs>
```

Listing C.6: Description of a district.xml file.

Routes In this file the interesting elements are defined:

- Vehicle and their Types
- Routes

For the creation of large amount of traffic a flow should be defined as well. A flow automatically generates vehicles.

In order to run the SUMO environment with defined vehicles a *route.xml* is needed which defines the routes of the vehicles in the simulation. The rou.xml can be created in two ways. One approach is shown at `http://sumo-sim.org/userdoc/Tutorials/Quick_Start.html` where the routes are defined manually. The other way is to generate the chosen route with a defined *OD-Matrix* (Origin to Destination). See section C.2.1 to create an OD-Matrix. When the OD-Matrix and its districts are defined, the trips.xml can be created as follows: **¿¿ od2trips –od-matrix-files=fileName_od_matrix.txt –net-file=fileName.districts.xml –output-file=fileName.trips.xml –timeline.day-in-hours=true**

Vehicle Type Vehicle types is defined through attributes. All attributes start with small letters.

Example:

```
<vType id="normalCar" accel="4.0" decel="4.0" length="6.0"
minGap="5.0" maxSpeed="10.0" speedFactor="1.3" tau="1.5"
sigma="0.5" color="137,137,137"/>
```

Attributes:

- id: unique id of the car type
- accel: acceleration of the car
- decel: breaking power of the car
- length: the longest side of a car
- minGap: minimal distance to the front car
- maxSpeed: maximum speed a car can drive

- speedFactor: in reality every driver has an own opinion about speed limits. With this factor the driver will drive 1.3 times as fast as the current speed limit of the edge.
- tau: not every driver has a fast reaction. With tau you define a delay in seconds. Only positive numbers are allowed!
- sigma: the drivers imperfection. How good/how tight does the driver stick to the parameters? (0-1)
- color: rgb color definition

SUMO itself has defined some default values for vehicles. They can be found here: `http://sumo-sim.org/userdoc/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html#Vehicle_Types`.

The SUMO probability values must be 1.0. NOTE: MATI does not support flow definitions!!!

Using SUMO

The Name.net.xml file contains the network and is ready for use in SUMO. This file is generated using the NETCONVERT-tool. It needs the edge-file, node-file and type-file if defined. The result is the network file Name.net.xml whose name has to be defined manually when using the tool via command line. Any changes here should be avoided. If to change something in the network, then generate this file again! It keeps all files consistent.

The Name.sumocfg.xml is the configuration file for SUMO.

Example:

```xml
 <?xml version="1.0" encoding="UTF-8"?>
<input>
    <net-file value="janaScen.net.xml"/>
    <route-files value="janaScen.rou.xml"/>
</input>
<time>
    <begin value="0"/>
    <end value="10000"/>
</time>
  <processing>
    <time-to-teleport value="-1"/>
</processing>
```

Listing C.7: Configuration File.

Sources and more information:

- `http://sumo-sim.org/userdoc/Tutorials/Quick_Start.html`
- `http://sumo-sim.org/userdoc/Networks/SUMO_Road_Networks.html`
- `http://sumo-sim.org/userdoc/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html#Default_Vehicle_Type`
  Link to MATI-scenario in Git:
- `mati/src/main/resources/scenarios/janaScenario/environmentFiles`

Vehicle connects to SUMO

Every agent has its own vehicle artifact through which the agent is connected to the SUMO vehicle. Perceptions from the SUMO environment are mapped to the agent believes with a detailed listing .

```
public class VehicleArtifact extends Artifact {
@OPERATION
public void init(HashMap<String, Object> perMap) {
this.agName = getId().getCreatorId().getAgentName();
for (String key : perMap.keySet()) {
if (key.equals("route")) {
route = (List<String>) perMap.get(key);
defineObsProperty("route", route.toString());
}
if (key.equals("speed")) {
speed = (Double)perMap.get(key);
defineObsProperty("speed", speed);
}
...
}

@OPERATION
public void shareRoute() { ... }

@OPERATION
public void changeRoute(ArtifactId arId) { ... }

@INTERNAL_OPERATION
public void updatePercepts(HashMap<String, Object> perMap) {
for (String key : perMap.keySet()) {
if (key.equals("route")) {
List<String> t_route = (List<String>) perMap.get(key);
if (route.equals(t_route)) {
break;
}
route = (List<String>) perMap.get(key);
getObsProperty("route").updateValue(route.toString());
}
if (key.equals("speed")) {
speed = (Double) perMap.get(key);
getObsProperty("speed").updateValue(speed);
}
...
}
}
```

Listing C.8: Vehicle Artifact connects to SUMO vehicle.

Interface TraCI4J

TraCI4J is an open-source project and is hosted on github.

Queries in TraCI4J This Subsection describes how to add queries to alter states of a TraCI object. Although the original TraCI (C++-version) protocol has many commands, not all of these are implemented in Traci4J. Since the MATI project is depending on this library, the sources need to be checked out in order to modify, translate and deploy the project by your own.

General structure The objects in TraCI4J are defined through XML-files, which can be modified instead of the JAVA Beans. After modification the ant-script ant generate is called to update the JAVA files. Since not all JAVA files are automatically generated, they are located in a folder called src/java-gen.

The XML files contain the name and the return values of the queries.

For example in the Vehicle.xml the queries are located:

```
<traciClass>
    <name>Vehicle</name>
       <readQueries>
         // read queries
       </readQueries>
       <changeStateQueries>
         // change state queries
       </changeStateQueries>
</traciClass>
```

Example: Adding a query to change the lane index

For setting an attribute of a TraCI object the corresponding changeState-Query needs to be called. If not already existing, this query needs to be added first. For example: to change the lane index write:

```
<changeStateQuery>
    <name>ChangeLaneIndex</name>
    <query>ChangeLaneIndexQuery</query>
    <affects>
      <affect>ReadCurrentLaneIndex</affect>
    </affects>
</changeStateQuery>
```

After a state was changed, it needs to be updated. As effect the read-method needs to be invoked. Therefore it is necessary to add a new read-query:

```
<readQuery>
   <name>ReadCurrentLaneIndex</name>
   <enum>LANE_INDEX</enum>
   <const>it.polito.appeal.traci.protocol.Constants.
          VAR_LANE_INDEX</const>
   <query>ReadObjectVarQuery.IntegerQ</query>
   <returnType>java.lang.Integer</returnType>
   <dynamic>true</dynamic>
</readQuery>
```

The enum is only used to retrieve the getter in the code via getRead-Query(LANE_INDEX). Next the variable id in the const field needs to be added. The constant contains the byte that is send over network to represent that variable. Luckily, the author of TraCI4J has already implemented these fields.

Implementing the Method The XML file only defines the method name, its return type and so on. A new ChangeQuery needs to be created. For our example the JAVA file ChangeLaneIndexQuery.java is created in the source folder. Existing queries can be used as template (since they are very much alike).

```
package it.polito.appeal.traci;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import de.uniluebeck.itm.tcpip.Storage;
import it.polito.appeal.traci.protocol.Constants;

public class ChangeLaneIndexQuery extends
ChangeObjectVarQuery<LaneIndexQueryParameter> {
 ChangeLaneIndexQuery(DataInputStream dis,
 DataOutputStream dos, String objectID ) {
  super(dis, dos, Constants.CMD_SET_VEHICLE_VARIABLE,
  objectID, Constants.CMD_CHANGELANE);
 }
 @Override
 protected void writeValueTo(LaneIndexQueryParameter buffer,
 Storage content) {
  content.writeByte(Constants.TYPE_COMPOUND);
  content.writeInt(2);
  content.writeByte(Constants.TYPE_BYTE);
  content.writeByte(buffer.getLaneIndex());
  // lane index
  content.writeByte(Constants.TYPE_INTEGER);
  content.writeInt(buffer.getDuration());
  // duration - time spent on lane
    }
}
```

Listing C.9: Implementation of Change Query.

The arguments in the super constructor are very important. They will be send over network (header) and tell the server what command this is and what parameters are following. The parameters (if any) are defined in the writeValueTo-method. The commands are already defined in it.polito.appeal.traci.protocol.Constants. However the parameters for each command needs to be placed, it states in the TraCI documentation `http://sumo.sourceforge.net/doc/current/docs/userdoc/TraCI.html#TraCI_Commands`.

Query use In the above example a new query was added to the vehicle object. It is now ready to be used.

```
public void changeLaneIndex(String vID, Integer laneIndex)
throws NullPointerException {
 try {
      LaneIndexQueryParameter param =
      new LaneIndexQueryParameter((byte)laneIndex, 500);
      Vehicle vehicle = vRepo.getByID(vID);
      if(null==vehicle) {
      throw new NullPointerException("Vehicle with ID
      " + vID + " is not in Repository!");}
      ChangeLaneIndexQuery query =
      vehicle.queryChangeLaneIndex();
      query.setValue(param);
      query.run();}
      catch( IOException e) {
       e.printStackTrace(); }
}
```

Listing C.10: Query Use.

Randomization of Vehicle Distribution

The simulation runs need to be randomized for statistic relevant statements. Here is an example of an own random implementation:

```
package mj_ia;

import jason.JasonException;
import jason.asSemantics.DefaultInternalAction;
import jason.asSemantics.InternalAction;
import jason.asSemantics.TransitionSystem;
import jason.asSemantics.Unifier;
import jason.asSyntax.NumberTerm;
import jason.asSyntax.NumberTermImpl;
import jason.asSyntax.Term;

import java.util.Random;

/**
 * Own random implementation
 * <p/>
 * - value is between [low,high[
/
public class randomInt extends DefaultInternalAction{
private static InternalAction singleton = null;
public static InternalAction create() {
  if (singleton == null)
  singleton = new randomInt();
  return singleton; }
  Random m_rand = new Random();
@Override
 public Object execute(TransitionSystem ts, Unifier un,
 Term[] args) throws Exception{
    int l_low;
    int l_high;
    if (args.length != 3)
    throw new JasonException("wrong number of args.");
    try{
      l_low = (int) ((NumberTerm) args[1]).solve();
      l_high = (int) ((NumberTerm) args[2]).solve();}
    catch (ClassCastException e){
    throw new JasonException
     ("low and high must be a number equal and greater 0");}
     if (l_low < 0 || l_high < 0)
    throw new JasonException
     ("low and high must be a number equal and greater 0");
     if (l_low >= l_high)
    throw new JasonException
     ("low high must be a lower than high");
    return un.unifies(args[0],
    new NumberTermImpl((int) (Math.random() *
     (l_high - l_low) + l_low)));
    }
}
```

scenario.json

```
{
    "coreSpecification": {
        "debug": false,
        "skippedSteps": 0,
        "stepDelay": 100
    },
    "scenarios": [
        {
            "environments": [
                {
                    "config": "./environmentFiles/sumoenv.json",
                    "file": "<relative path from scenario file>
                    /sumoenvironment-1.0-SNAPSHOT-jar-with-dependencies.jar",
                    "id": "1"
                }
            ],
            "interpreters": [
                {
                    "config": "./interpreterFiles/interpreter.json",
                    "file": "<relative path from scenario file>
                    /<Interpreter Jar>",
                    "id": "1",
                    "usedEnvID": "1"
                }
            ],
            "start": true,
            "steps": 100
        }
    ]
}
```

Environment configuration (sumoenv.json)

Not that: *The configuration is dynamically build by the shell script and does not be build by hand.*

Interpreter configuration (interpreter.json)

configuration for the interpreter **(structure unknown)**

Tool configuration (tool.json)

configuration for tools like HDF5 **(structure unknown)**

SUMO net configuration (build directory)

Within the environmentFiles directory is the subdirectory "build" with all SUMO relevant data which is mandatory for the configuration generation. The file "build.xml" creates the configuration for the SUMO netconverter (`http://sumo-sim.org/userdoc/NETCONVERT.html`). Especially the output file in the configuration needs the following tag:

```
    <output>
        <output-file value="../net.xml"/>
    </output>
```

This file always points to the subordinate directory ("environmentFiles") and there to the file "net.xml"!

Execution

To execute MATI including SUMO, MATI needs to be build with mvn install. MATI is started with the following steps:

```
- if required build MATI
- scenario is called with runmati.bash (Linus / OSX)
or runmati.bat (Windows)
- runmati.bat / runmati.bash
<target/scenarios/<ScenarioDirectory>/scenario.json
```

C.2.2 Analyzing Tools

MATI provides an interface 'ITool' for the performance tools so any tool can be connected.

Simple Tool

Simple Tool is self-created and in the described in the JAVA class C.12.

```
package mati.tools.matisimpletool;
import apltkex.common.core.CCore;
import apltkex.common.core.CEnvironmentStepResult;
import apltkex.common.core.CInterpreterStepResult;
import apltkex.common.interpreter.IInterpreter;
import apltkex.common.specification.CComponentSpecification;
import apltkex.common.tool.EXTool;
import apltkex.common.tool.ITool;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.data.general.Dataset;
import org.jfree.data.time.DynamicTimeSeriesCollection;
import org.jfree.data.time.Second;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.ui.ApplicationFrame;
import java.awt.*;
import java.util.Collection;
import java.util.Date;
import java.util.Random;
public class CNumAgentTool implements ITool{
 private String toolTitle = null;
 private ApplicationFrame m_frame = null;
 private static final int COUNT = 2 * 60;
 private static final float MINMAX = 100;
 private static final String LABEL_X_AXIS_AgentsChart =
 "Cycle";
 private static final String LABEL_Y_AXIS_AgentsChart =
 "Number of Agents";
 private static int numberOfAgents = 0;
 private Dataset datasetAgents;
 private JFreeChart chartAgents;
 private Random randomGenerator = new Random();
 @Override
  public void init(CComponentSpecification p_comSpec)
   throws EXTool{
   String windowTitle = p_comSpec.getId();
   this.toolTitle = windowTitle;
   m_frame = new ApplicationFrame(windowTitle);
// generate some test data
   datasetAgents = createXYDataset();
   chartAgents = createXYLineChart((XYDataset)
   datasetAgents, "Number of Agents");
// place chart in center
   ChartPanel panelAgents = new ChartPanel(chartAgents);
   m_frame.add(panelAgents, BorderLayout.CENTER);
// resize window
   m_frame.setPreferredSize(new Dimension(800, 600));
   m_frame.setSize(800, 600);
// set frame
   m_frame.setVisible(true);
}
 @Override
  public void processInterpreterStepResult
  (Collection<CInterpreterStepResult> stepResults){
   for (CInterpreterStepResult result : stepResults){
   IInterpreter interpreter = result.interpreter;
   numberOfAgents = interpreter.getNmbOfAgents()
    updateNumberOfAgents(numberOfAgents); }
}
 @Override
  public void processEnvironmentStepResults
  (Collection<CEnvironmentStepResult> stepResults) {
//For changes use File | Settings | File Templates. }
 @Override
  public void release(){
    m_frame.dispose();}
 @Override
  public void setCore(CCore core) {
//For changess use File | Settings | File Templates. }
 @Override
  public String getName() {
  return toolTitle;
//For changes use File | Settings | File Templates. }
public DynamicTimeSeriesCollection
  createTimeSeriesDataset() {
  DynamicTimeSeriesCollection collection =
  new DynamicTimeSeriesCollection(1, COUNT, new Second());
  collection.setTimeBase(new Second(new Date()));
  collection.addSeries(gaussianData(), 0, "Cycles");
  return collection;}
public XYSeriesCollection createXYDataset() {
  final XYSeries series = new XYSeries("Agents");
  series.add(0.0, 0);
  XYSeriesCollection collection = new XYSeriesCollection(series);
  return collection;}
public JFreeChart createXYLineChart(XYDataset dataset, String title){
  JFreeChart chart = ChartFactory.createXYLineChart(
  title, LABEL_X_AXIS_AgentsChart, LABEL_Y_AXIS_AgentsChart,
  dataset, PlotOrientation.VERTICAL, true, true, false);
  final XYPlot plot = (XYPlot) chart.getPlot();
  ValueAxis domain = plot.getDomainAxis();
  domain.setAutoRange(true);
  ValueAxis range = plot.getRangeAxis();
  range.setRange(0, MINMAX);
// white background and black grindlines
  plot.setBackgroundPaint(Color.white);
  plot.setDomainGridlinesVisible(true);
  plot.setRangeGridlinesVisible(true);
  plot.setRangeGridlinePaint(Color.black);
  plot.setDomainGridlinePaint(Color.black);
  return chart; }
private void updateNumberOfAgents(float number){
 if (datasetAgents instanceof DynamicTimeSeriesCollection){
   float[] newData = new float[1];
   newData[0] = number;
  ((DynamicTimeSeriesCollection) datasetAgents).advanceTime();
  ((DynamicTimeSeriesCollection) datasetAgents).appendData(newData); }
 else if (datasetAgents instanceof XYSeriesCollection) {
   double yValue = number;
   XYSeries series = ((XYSeriesCollection)
   datasetAgents).getSeries("Agents");
   series.add(series.getItemCount(), yValue); }
 }
private float randomValue(){
 return (float) (randomGenerator.nextGaussian() * MINMAX / 3);}
private float[] gaussianData(){
 float[] a = new float[COUNT];
  for (int i = 0; i < a.length; i++) {
// a[i] = randomValue();
   a[i] = 0; }
  return a; }
public static void main(String[] args){
   CNumAgentTool st = new CNumAgentTool();
```

```
    try {
         st.init(null);}
    catch (Throwable e){
         e.printStackTrace();} }
}
```

Listing C.11: MATI Class Simple Tool.

The Performance Tool measures the MATI performance presented in Chapter 4.3.5.

```
package mati.tools.performancetool;
import apltkex.common.core.CCore;
import apltkex.common.core.CEnvironmentStepResult;
import apltkex.common.core.CInterpreterStepResult;
import apltkex.common.specification.CComponentSpecification;
import apltkex.common.tool.EXTool;
import apltkex.common.tool.ITool;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.data.general.Dataset;
import org.jfree.data.time.DynamicTimeSeriesCollection;
import org.jfree.data.time.Second;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.ui.ApplicationFrame;
import java.awt.*;
import java.util.Collection;
import java.util.Date;
import java.util.Random;
public class CPerformanceTool implements ITool{
 private String toolTitle = null;
 private ApplicationFrame m_frame = null;
 private static final int COUNT = 2 * 60;
 private static final String LABEL_X_AXIS_CycleChart = "Time";
 private static final String LABEL_Y_AXIS_CycleChart = "Cycles";
 private static final float MINMAX = 100;
 private static long lastNumOfCycles = 0;
 private long lastUpdate = 0;
 private Dataset datasetCycles;
 private JFreeChart chartCycles;
 private Random randomGenerator = new Random();
@Override
 public void init(CComponentSpecification p_comSpec)
 throws EXTool{
  this.toolTitle = p_comSpec.getId();
  m_frame = new ApplicationFrame(toolTitle);
// generate some test data
  datasetCycles = createTimeSeriesDataset();
  chartCycles = createTimeSeriesChart((DynamicTimeSeriesCollection)
  datasetCycles, "Cycles per second");
// place chart in center
  ChartPanel panelCycles = new ChartPanel(chartCycles);
  m_frame.add(panelCycles);
// resize window
  m_frame.setPreferredSize(new Dimension(800, 600));
  m_frame.setSize(800, 600);
// set frame
   m_frame.setVisible(true); }
@Override
 public void processInterpreterStepResults(
  Collection<CInterpreterStepResult> stepResults){
  for (CInterpreterStepResult result : stepResults) {
// check if cycles per second can be updated
  long timeDelta = System.currentTimeMillis() - lastUpdate;
  if (timeDelta > 1000){
  System.out.println("Printing steps : " + result.step);
  updateCyclesPerSecond(result.step);
  lastUpdate += timeDelta;}
  }
 }
@Override
 public void processEnvironmentStepResults(
 Collection<CEnvironmentStepResult> stepResults){}
@Override
 public void release(){
  m_frame.dispose(); }
@Override
 public void setCore( CCore core ){}
@Override
 public String getName(){
  return toolTitle; }
 private DynamicTimeSeriesCollection
 createTimeSeriesDataset(){
  DynamicTimeSeriesCollection collection = new DynamicTimeSeriesCollection
  (1, COUNT, new Second());
  collection.setTimeBase(new Second(new Date()));
  collection.addSeries(gaussianData(), 0, "Cycles");
  return collection;}
private XYSeriesCollection createXYDataset(){
  final XYSeries series = new XYSeries("Agents");
  series.add(1.0, 0);
  series.add(2.0, 5);
  series.add(3.0, 10);
  XYSeriesCollection collection = new XYSeriesCollection(series);
  return collection;}
private JFreeChart createTimeSeriesChart
  (XYDataset dataset, String title) {
  JFreeChart chart = ChartFactory.createTimeSeriesChart(
  title, LABEL_X_AXIS_CycleChart, LABEL_Y_AXIS_CycleChart,
  dataset, true, true, false);
  final XYPlot plot = (XYPlot) chart.getPlot();
  ValueAxis domain = plot.getDomainAxis();
  domain.setAutoRange(true);
  ValueAxis range = plot.getRangeAxis();
  range.setRange(0, MINMAX);
// white background and black grindlines
  plot.setBackgroundPaint(Color.white);
  plot.setDomainGridlinesVisible(true);
  plot.setRangeGridlinesVisible(true);
  plot.setRangeGridlinePaint(Color.black);
  plot.setDomainGridlinePaint(Color.black);
  return chart; }
private void updateCyclesPerSecond(long cycles){
//add new item
  float[] newData = new float[1];
  newData[0] = cycles - lastNumOfCycles;
  ((DynamicTimeSeriesCollection) datasetCycles).advanceTime();
  ((DynamicTimeSeriesCollection) datasetCycles).appendData(newData);
  lastNumOfCycles = cycles; }
private float randomValue(){
  return (float) (randomGenerator.nextGaussian() * MINMAX / 3);}
private float[] gaussianData(){
  float[] a = new float[COUNT];
  for (int i = 0; i < a.length; i++)
    a[i] = 0;
    return a; }
}
```

Listing C.12: Class Performance Tool.

HDF5

HDF 5 is available at `https://www.hdfgroup.org/HDF5/` accessed 15.12.2015:

> Current Release: HDF5-1.8.16
>
> HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.

```
package mati.tools.groupingbenchmarktool;
// without CamelCase so jsonParser can associate attributes
public class CGroupingBenchmarkToolSpecification{
    private int m_nmbofagents;
    private String m_hdffilepath = "./output.h5";
    private String m_grouppath = "/default";
//////////////////////////////////////////////////////////
// Getter
//////////////////////////////////////////////////////////
public int getNmbofagents(){
 return m_nmbofagents;}
public String getHdffilepath() {
 return m_hdffilepath; }
public String getGrouppath() {
 return m_grouppath; }
/////////////////////////////////////////////////////////
// Setter
/////////////////////////////////////////////////////////
public void setNmbofagents(int p_nmbOfAgents) {
 m_nmbofagents = p_nmbOfAgents; }
public void setHdffilepath(String p_hdffilepath) {
 m_hdffilepath = p_hdffilepath;}
public void setGrouppath(String p_grouppath){
 m_grouppath = p_grouppath;}
}
```

Listing C.13: MATI Grouping Benchmark Tool Specification.

This is the MATI tool in current use.

```
package mati.tools.groupingbenchmarktool;

import apltkex.common.core.CCore;
import apltkex.common.core.CEnvironmentStepResult;
import apltkex.common.core.CInterpreterStepResult;
import apltkex.common.specification.CComponentSpecification;
import apltkex.common.specification.CSpecificationParser;
import apltkex.common.specification.EXScenarioParser;
import apltkex.common.tool.EXTool;
import apltkex.common.tool.ITool;
import apltkex.common.util.CHDF;
import apltkex.common.util.CPosition;
import apltkex.common.util.EXHDF;
import mati.interpreters.grouping.CGroupingAgArch;
import mati.interpreters.grouping.CGroupingInterpreter;
import mati.interpreters.grouping.CSimilarCalculator;
import java.io.File;
import java.util.Collection;
import java.util.Map;
public class CGroupingBenchmarkTool implements ITool{
 public final String PARAM_TABLES = "/paramstables/";
 public final String DIS_TABLES   = "/distables/";
 public final String GROUP_TABLES = "/groupingtables/";
 public final static int NMB_OF_PARAM = 14;
 private int m_nmbOfAgents;
 private long chunk_dims[] = {1, 1};
 private String m_hdfFilePath;
 private String m_preferredGroupPath;
 private String m_groupPath = null;
 CHDF m_hdfFile = null;
 private double[][] l_agDisTable;
 private double[][] l_agParamTable;
 private int[][] l_agGroupTable;
 private CCore m_core = null;
@Override
 public void init(CComponentSpecification p_toolSpec)
 throws EXTool { try {
  CGroupingBenchmarkToolSpecification l_gbts =
  CSpecificationParser.parse(
  new File(m_core.getBasePath() + p_toolSpec.getConfig()),
  CGroupingBenchmarkToolSpecification.class );
 System.out.println("^^^^^^^^^^^^^^^^^^^^^^^^ number of agents:
 " + l_gbts.getNmbofagents());
 m_nmbOfAgents = l_gbts.getNmbofagents();
 m_hdfFilePath = l_gbts.getHdffilepath();
 m_preferredGroupPath = l_gbts.getGrouppath();
 m_hdfFile = new CHDF(m_hdfFilePath);
 m_groupPath = getGroupId(m_hdfFile, m_preferredGroupPath);
 l_agDisTable   = new double[m_nmbOfAgents][m_nmbOfAgents];
 l_agParamTable = new double[m_nmbOfAgents][NMB_OF_PARAM];
 l_agGroupTable = new int[m_nmbOfAgents][1];
 chunk_dims[0] = m_nmbOfAgents; } catch (EXScenarioParser e){
   throw new EXTool(EXTool.EError.PARSE, e);}
   catch (EXHDF exhdf) {
    exhdf.printStackTrace(); }
    catch (Exception e) {
    e.printStackTrace();}
  }
@Override
 public void processInterpreterStepResults
 (Collection<CInterpreterStepResult> p_stepResults){
 int l_agId;
 CPosition l_pos,l_des;
 double[] l_simFuncArray;
 for (CInterpreterStepResult l_intStepResult : p_stepResults){
 CGroupingInterpreter l_interpreter =
 (CGroupingInterpreter) l_intStepResult.interpreter;
 CSimilarCalculator l_calculator = l_interpreter.getSimilarCalculator();
//clear tables
  for(int l_i = 0 ; l_i < m_nmbOfAgents; l_i++)
   for(int l_j = 0 ; l_j < m_nmbOfAgents; l_j++)
       l_agDisTable[l_i][l_j] = 0;
  for(int l_i = 0 ; l_i < m_nmbOfAgents; l_i++)
```

```
   for(int l_j = 0 ; l_j < NMB_OF_PARAM; l_j++)
       l_agParamTable[l_i][l_j] = 0;
  for (int i = 0; i < m_nmbOfAgents; ++i)
   for (int j = 0; j < 1; ++j)
     l_agGroupTable[i][j] = 0;
  for (CGroupingAgArch l_agArch : l_interpreter.getAllAgArchs()) {
   l_agId = l_agArch.getId();
   l_pos = l_agArch.getPosition();
   l_des = l_agArch.getDestination();
   Map<CGroupingAgArch, Double> l_similarMap =
   l_calculator.getSimilarMap(l_agArch);
// create DistanceTable
  for(Map.Entry<CGroupingAgArch, Double> l_entry:
   l_similarMap.entrySet()){
    if(l_entry.getValue() == null)
       System.out.println("Break");
    if(l_agArch.getId() == 6 && l_entry.getKey().getId() == 22)
       System.out.println("Test: "+ l_entry.getValue());
       l_agDisTable[l_agId][l_entry.getKey().getId()] =
       l_entry.getValue();
//System.out.println("value of distable: " + l_entry.getValue());
//System.out.println("type of distable value: " + l_entry.getClass().getName());
}
// create paramTable
 /**
 * @author Thomas Hornoff (thomas.hornoff@tu-clausthal.de)
 *
 * values [0]  - [5]    -> alpha values see: CAEDF.java in mj_func
 * values [6]  - [9]    -> current position [6] & [7],
 *              destination [8] & [9]
 * values [10] - [13]   -> clear
 * only double values are allowed!!!
 */
 l_simFuncArray = l_agArch.getSimFunc().getParams();
 l_agParamTable[l_agId][0] = l_simFuncArray[0];
 l_agParamTable[l_agId][1] = l_simFuncArray[1];
 l_agParamTable[l_agId][2] = l_simFuncArray[2];
 l_agParamTable[l_agId][3] = l_simFuncArray[3];
 l_agParamTable[l_agId][4] = l_simFuncArray[4];
 l_agParamTable[l_agId][5] = l_simFuncArray[5];
 l_agParamTable[l_agId][6] = l_pos.m_x;
 l_agParamTable[l_agId][7] = l_pos.m_y;
 l_agParamTable[l_agId][8] = l_des.m_x;
 l_agParamTable[l_agId][9] = l_des.m_y;
 l_agParamTable[l_agId][10] = l_agArch.getSpeedMax();
 l_agParamTable[l_agId][11] = l_agArch.getAccel();
 l_agParamTable[l_agId][12] = l_agArch.getDecel();
 l_agParamTable[l_agId][13] = l_agArch.getSpeed();
// create groupingTable
// Stores the group id of every agent.
// Therefore, agents with the same id are in one group.
 if (l_agArch.getGroup() != null) {
 l_agGroupTable[l_agId][0] =
 l_agArch.getGroup().getGroupID();} else {
 l_agGroupTable[l_agId][0] = 0;}
}
try {
  m_hdfFile.store(m_groupPath + DIS_TABLES   + l_intStepResult.step,
  l_agDisTable,   chunk_dims);
  m_hdfFile.store(m_groupPath + PARAM_TABLES + l_intStepResult.step,
  l_agParamTable, chunk_dims);
  m_hdfFile.store(m_groupPath + GROUP_TABLES + l_intStepResult.step,
  l_agGroupTable, chunk_dims); }
  catch (Exception e) {
         e.printStackTrace(); }
}
//  m_agDisTableList.add(l_agDisTable);
//  m_agParamTableList.add(l_agParamTable);
    }
@Override
 public void processEnvironmentStepResults
 (Collection<CEnvironmentStepResult>
 p_stepResults){}
@Override
 public void release() throws EXTool {
  try {
// CHDF l_hdfFile = new CHDF(m_hdfFilePath);
// String l_groupId = getGroupId(l_hdfFile, m_preferredGroupPath);
// m_hdfFile.store(m_groupPath + PARAM_TABLES, m_agParamTableList);
// l_hdfFile.store(l_groupId + DIS_TABLES, m_agDisTableList);
      } catch (Exception e){
        throw new EXTool(EXTool.EError.RELEASE,e); }
    }
@Override
 public void setCore(CCore p_core){
   m_core = p_core; }
@Override
 public String getName(){
 return null;}
// group id is the name of the current scenario, e.g. '/hildesheimer'
 private String getGroupId(CHDF p_hdfFile, String p_preferredId)
 throws EXHDF {
     if(!p_hdfFile.containGroup(p_preferredId))
        return p_preferredId;
        String l_availableId;
        int l_index = 0;
        while (true) {
        l_availableId = p_preferredId + "_" + ++l_index;
        if(!p_hdfFile.containGroup(l_availableId))
        return l_availableId;}
    }
}
```

Listing C.14: MATI Grouping Benchmark Tool.

MATI uses the SimpleTool for first logging, the PerformanceTool for measuring MATI performance and the Grouping Benchmark Tool for the Evaluation Experiments in the following Chapter 7.

C.3 MATI Test

C.3.1 Scenario Definition

The evaluation scenario is a grid with a size of 20×20 vertices's, the edges are streets with 5 lanes (in each direction) and a length of $400m$. The route of each vehicle is randomly generated. Each vertex is a source for new vehicles. This scenario should minimize the possibility of a traffic jam, which can arise by accumulation during the generating process of vehicles. All net data is generated in a preprocessing run before the simulation is started. The initialization of the vehicle start points should be hence independent and identically distributed. SUMO is used for the traffic simulation and the SUMO configuration (net, routing and initialization node) is generated in a preprocessing run.

C.3.2 Results

The experiments for each agent type are run on two machines; the simulation parts are split into disjoint sets, so that each machine receives approximately equal work load. The experiments start around 4:30pm (cf. memory and CPU utilization). To avoid the Java garbage processing call, the memory allocation is increased and the simulation restarted on each increment step.

Hello World Agent: The *Hello World agent* experiment starts with around 60% of CPU utilization and 30% of memory utilization. The CPU work load is reduced after a short time, because the hello agents creates one message output only. The work load can expand up to 85% in later tests by including more agents and vehicles. The CPU work load drops down.

Figure C.3 shows a small bias of the result distribution with a large interquartile range (25- and 75-quartile). Median and average are almost constant, which is explainable by the structure of the agent. The agent runs only one logging call and after the call only SUMO is used (to move the vehicle).

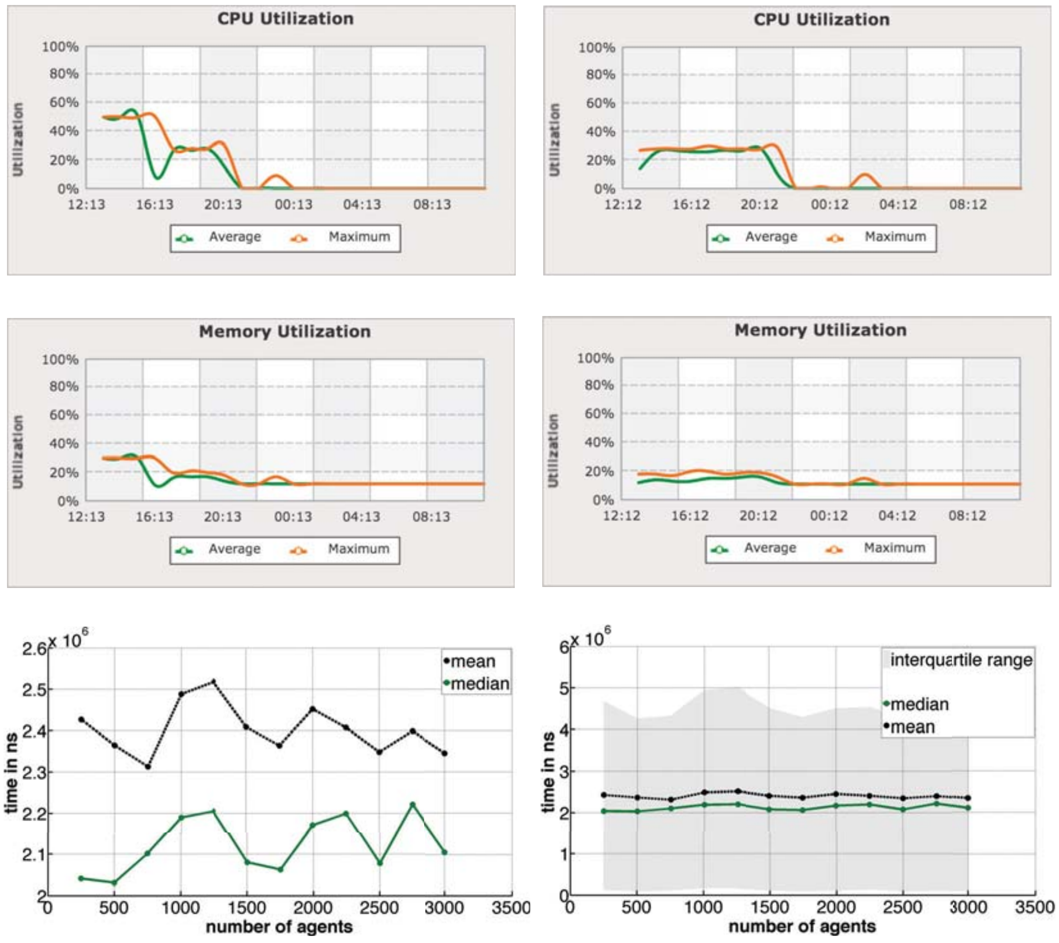


Figure C.3: CPU, Memory Utilization and Execution Time – Hello World Agent.

Fibonacci Agent: Figure C.4 shows the results for the *Fibonacci agent*. The starting conditions of this run are mostly equal. The CPU utilization is almost constant during the simulation run. The agent cycle execution time is a little bit larger, because the Fibonacci agent need some small integer calculation. In a global view the time complexity is constant. As related to the system load the calculation is not complex enough to create significant values.

The experiments show in many cases expected results. Execution time and CPU work load correspond with the agent type. An noteworthy issue is the large interquartile rate (IQR). The guess is that Java Bean component creates the large IQR, which run a firePropertyChange to inform the GUI element on changes. On a short test with removing some of these calls, the IQR could be reduced around the factor 10. MATI uses a large number of these calls; in the internal structures, a code refactoring is needed to remove all calls.

The CPU work load of the Hello agent shows a fast decrease, because the agents log their message and after this call SUMO must update the position within the grid only. Routing and other algorithms / data are cached by the

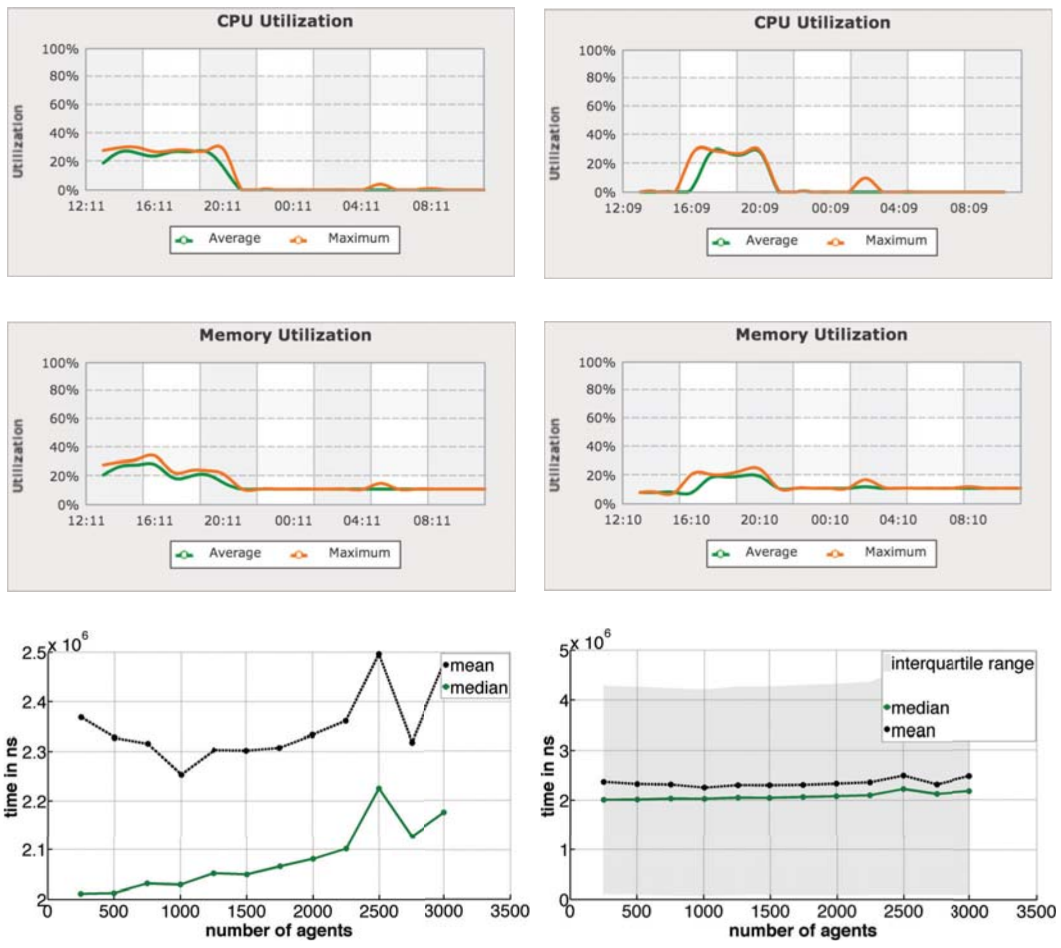


Figure C.4: CPU, Memory Utilization, and Execution Time – Fibonacci Agent.

SUMO environment, so there is no need to create more work load. Median and average of the execution time are similar, so the bias and the adjustment of the result's distribution are within an acceptable range.

The Fibonacci agent scenario returns a similar result, but the difference is the execution time graph. The variances between the two scenario are small, so there cannot be a more detailed interpretation.

These first test scenarios point out a working alpha framework. With a code refactoring process the framework can create better (detailed) results. In this case more complex scenarios are needed with more complex agents e.g. communication between agents is not tested for valid statements, but is possible. The complexity of an agent should extend with information about the world (environment), so the agent stores its local view of the current street/ lane data and the neighborhood vehicles. On this information communication structure can be created between the neighbors.

C.3.3 MATI Groups

The main sequence of the grouping algorithm starts at mati/mati/apltkexcommon/src/main/java/apltkex/common/CApp.java and is as follows.

- main() → start/run thread
- CCore → doInterpreters → run → IInterpreter.step()
- CGroupingInterpreter → step-method → CSimilarCalculator.step()
- CArtifact.step() → CGroupingAgArch.step() → reasoningCycle

Tools process interpreter steps with

- CGroupingBenchmarkTool → process InterpreterStepResults() which fills distables, paramstables, and groupingtables.
- Agent coming in: CGroupingInterpreter.java notes agents coming in with handleNewEntity()
- Leaving agents: CGroupingInterpreter.java associates agents leaving with the function removeAgent()

Collaboration diagram in Figure C.5:

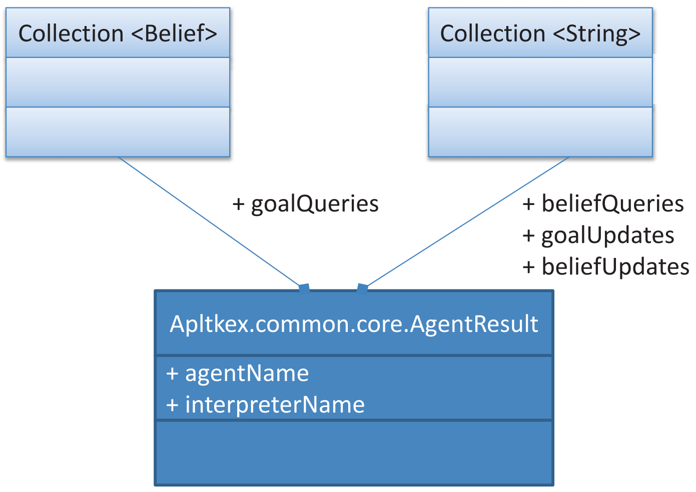


Figure C.5: Collaboration diagram for apltkex.common.core.AgentResult.

with public attributes:

- String agentName
- String interpreterName
- collection ⟨ String ⟩ beliefUpdates
- collection ⟨ String ⟩ beliefQueries
- collection ⟨ String ⟩ goalUpdates
- collection ⟨ Belief ⟩ goalQueries

C.3.4 Color Sort Implementation

C.3.5 CNP Implementation

The following example is from of the Jason sources [49]. It is used as basis for our cooperative MATI agents. Definition of the Multi-Agent-System:

```
\begin{algorithm}
MAS cnp {
infrastructure: Centralised

agents:
c; // the CNP initiator
p #3; // the participants (3) that offer a service
pr; // a participant that always refuses
pn; // a participant that does not answer
}
\end{algorithm}
```

Initiator

```
\begin{displaymath}
/* Initial beliefs and rules */

all_proposals_received(CNPId)

:- .count(introduction(participant,_),NP) & // number of participants
.count(propose(CNPId,_), NO) & // number of proposes received
.count(refuse(CNPId), NR) & // number of refusals received

NP = NO + NR.

/* Initial goals */
!startCNP(1,fix(computer)).

/* Plans */
// start the CNP
+!startCNP(Id,Task)
<- .print("Waiting participants...");
.wait(2000); // wait participants introduction
+cnp_state(Id,propose); // remember the state of the CNP
.findall(Name,introduction(participant,Name),LP);
.print("Sending CFP to ",LP);
.send(LP,tell,cfp(Id,Task));
// the deadline of the CNP is now + 4 seconds, so
// the event +!contract(Id) is generated at that time
.at("now +4 seconds", { +!contract(Id) }).

// receive proposal
// if all proposal have been received, don't wait for the deadline
@r1 +propose(CNPId,_Offer)
: cnp_state(CNPId,propose) & all_proposals_received(CNPId)
<- !contract(CNPId).

// receive refusals
@r2 +refuse(CNPId)
: cnp_state(CNPId,propose) & all_proposals_received(CNPId)
<- !contract(CNPId).

// this plan needs to be atomic so as not to accept
// proposals or refusals while contracting
@lc1[atomic]
+!contract(CNPId)

: cnp_state(CNPId,propose)
```

```
<- -+cnp_state(CNPId,contract);
.findall(offer(O,A),propose(CNPId,O)[source(A)],L);
.print("Offers are ",L);

L \== []; // constraint the plan execution to at least one offer
.min(L,offer(WOf,WAg)); // sort offers, the first is the best
.print("Winner is ",WAg," with ",WOf);
!announce_result(CNPId,L,WAg);
-+cnp_state(CNPId,finished).

// nothing todo, the current phase is not 'propose'
@lc2 +!contract(_).

-!contract(CNPId)
<- .print("CNP ",CNPId," has failed!").
+!announce_result(_,[],_).
// announce to the winner
+!announce_result(CNPId,[offer(_,WAg)|T],WAg)

<- .send(WAg,tell,accept_proposal(CNPId));
!announce_result(CNPId,T,WAg).
// announce to others
+!announce_result(CNPId,[offer(_,LAg)|T],WAg)
<- .send(LAg,tell,reject_proposal(CNPId));

!announce_result(CNPId,T,WAg).
\end{displaymath}
```

Participant

```
\begin{displaymath}
// gets the price for the product,
// a random value between 100 and 110.
price(_Service,X) :- .random(R) & X = (10*R)+100.
plays(initiator,c).

/* Plans */
// send a message to the initiator introducing myself as a participant
+plays(initiator,In)
: .my_name(Me)
<- .send(In,tell,introduction(participant,Me)).

// answer to Call For Proposal
@c1 +cfp(CNPId,Task)[source(A)]
: plays(initiator,A) & price(Task,Offer)
<- +proposal(CNPId,Task,Offer); // remember my proposal
.send(A,tell,propose(CNPId,Offer)).

@r1 +accept_proposal(CNPId)
: proposal(CNPId,Task,Offer)
<- .print("My proposal '",Offer,"' won CNP ",CNPId,
" for ",Task,"!").
// do the task and report to initiator
@r2 +reject_proposal(CNPId)
<- .print("I lost CNP ",CNPId, ".");
-proposal(CNPId,_,_). // clear memory
\end{displaymath}
```

MATI was tested with two different agent types. For a first short trial, to show the affects of agent code on the overall performance of the simulation. A simple agent was tested against a more complex one. In this case a *"Hello world"* and a *"Fibonacci"* agent.

The *"Hello world"* agent prints a message after it has been created and creates a log message. It has then achieved its only goal and is finished. The complexity of the *"Hello world"* agent is constant.

In comparison the *"Fibonacci"* agent calculates the Fibonacci sequence. It never finishes its goal. The numbers are calculated iteratively ad infinitum. The complexity of a Fibonacci agent highly depends on the implementation. This recursive agent has a linear complexity of $O(n)$. To calculate the Nth Fibonacci number it takes $N - 1$ runs to get the results and $O(n-1) = O(n)$.

The experiment started with 250 agents and was increased by 250 agents, in equidistant steps, until a maximum of 3000 agents. The metric definition uses the difference in system time (in nanoseconds) of two measuring point inside a method call. The statistics is created within the simulation, to avoid summarizing the constant value of function / method jumps.

C.4 Evaluation of Simulation

C.4.1 Structure of the HDF5 file

Inside the root directory of the hdf5 file the scenarios are stored, i.e. Southern part of Hanover (HS), three lanes, green wave (HI). Inside those scenarios groups there are stored two tables for each scenario, i.e. a *vehicle* and a *group* table. In the following Paragraphs the structure of these tables are defined including the characteristics of the stored data.

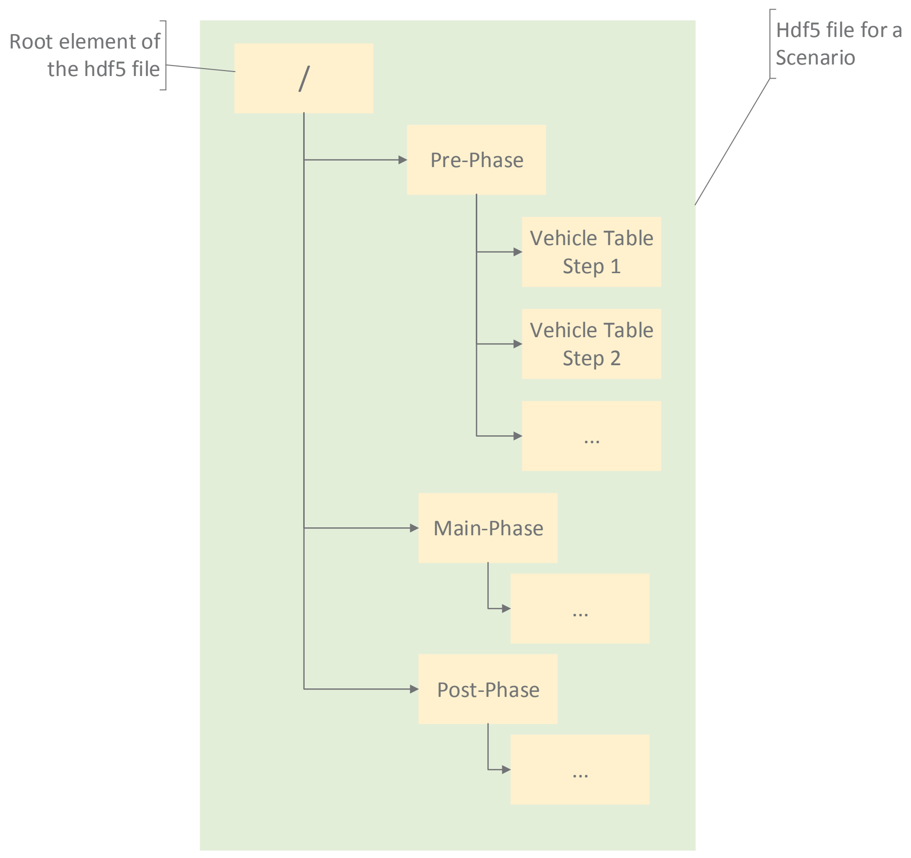


Figure C.6: Structure of a MDF5 file.

Figure C.6 shows the basic structure of the hdf5 file that is used to store simulated data of the scenarios and to use the hdf5 files to get the data into

MATLAB. Figure C.7 shows an example how to create an hdf5 file with Python. This Python file would create a *grueneWelle.h5* file including a table with 2000 rows and the defined columns. Additionally, each entry would be initialized with the value one *(np.ones)*. The file is created by running the Python script with *python myHDF5example.py* if myHDF5example.py was the Python file name.

```
import numpy as np
import tables as tb

f = tb.openFile('grueneWelle.h5', 'a')
grueneWelle = np.ones(2000, dtype=[('travelTime', float), ('holdTime', float),
                                   ('groupSize', int), ('position', float),
                                   ('speed', float), ('routeCost', int),
                                   ('initiator', bool), ('participant', bool),
                                   ('proposal', int), ('refusal', int), ('accept', int),
                                   ('reject', int), ('alpha1', float), ('alpha2', float),
                                   ('alpha3', float), ('alpha4', float),
                                   ('alpha5', float), ('total', float)])
f.createTable('/', 'grueneWelle', grueneWelle)
f.close()
```

Figure C.7: Example of creating a hdf5 file with Python.

Vehicle Table

The *Vehicle Table* stores relevant data for each vehicle running in a simulation with MATI. The following table specifies the values which must be stored in a table including their type. These values represent the columns inside the *Vehicle Table*. For every simulation step such a table is created.

Criteria for MATLAB Evaluation

The simulation is divided into three phases (P-M-A). The *preliminary phase* (P), the *main phase* (M), and the *after phase* (A).

Each scenario must run at least 10 times in order to compare and evaluate the results in MATLAB. There must be 5275 vehicles defined in the *Hannover-Suedstadt-Scenario* and 2723 vehicles in the *Gruene-Welle-Scenario*.

Filled Net and Phases Definition

Each scenario has to simulate the three phases, preliminary phase, main phase, and the after phase. Furthermore it has to be specified how many vehicles are

Name	**Type**
desiredSpeed	type of float; represents the desired speed of a vehicle in one step of one simulation-run.
acceleration	type of float; represents the current acceleration of a vehicle in one step of one simulation-run.
deceleration	type of float; represents the deceleration of a vehicle in one step of one simulation-run.
routeCost	type of float; represents the current route cost of a vehicle of one simulation-run.
journeyTime	type of float; represents the current journey time a vehicle of one simulation-step.
currentPosition	type of float; represents the the current position on the route of a vehicle of one simulation-step.
destination	type of float; represents the destination of a vehicle for each simulation-step.
alpha01	type of float; represents the alpha value of the desired speed.
alpha02	type of float; represents the alpha value of the acceleration.
alpha03	type of float; represents the alpha value of the deceleration.
alpha04	type of float; represents the alpha value of the route cost.
alpha05	type of float; represents the alpha value of the journey time.
alpha06	type of float; represents the alpha value of the current position.
alpha07	type of float; represents the alpha value of the destination.
total_MDF	type of float; represents the total value of the Manhattan Distance Function, i.e. alpha01 * desiredSpeed + alpha02 * acceleration + ...

Table C.1: Values and Types of the Vehicle Table which is stored each simulation step.

at the beginning in the scenario and where in order to evaluate different results of simulation runs.

Pre-phase At the beginning of a simulation all vehicles are in the preliminary phase meaning that they are performing independently and are trying to build a group with other vehicles. Since they are in a group, or have decided not to be in a group, the main phase of the simulation starts.

Main-Phase As soon as all vehicles are either in a group or driving independently the simulation progresses. The group behavior of vehicles is desired in order to maximize the performance and flow of the traffic simulation in order to push traffic flows.

Post-phase The moment vehicles fork a group the vehicle is entering the after phase. In that phase the vehicles are approaching their destinations.

Filled Net with IID IID stands for Independent, Identically Distributed and is a behavior of variables in statistics. Take the probability of tossing a coin. Each toss is *independent* since previous results do not influence the current result. Furthermore, it is *identically distributed* since chance are identical each toss.

This behavior is used to place the vehicles in the simulation at the beginning.

At the beginning of each scenario 100 cars are used with the distribution of heterogeneous vehicles described in Chapter 6.3.2).

C.4.2 Setup

In the following script determines the analysis of the preprocessed data. First, the workspace is emptied and subsequently the files to analyze are defined in the .mat format.

```matlab
% Clear all variables
clear
% Change working directory
cd '/Users/jgoermer/Desktop/Diss MATI Scenarios/MATI_Scenarios';
% Add function path
addpath('02_code');
% Define .mat files
files = ['heterogen_hildesheimer_15_go_1000_steps';
         'heterogen_suedstadt_15_go_1000_steps   ';
         %'homogen_grid_1_go_500_steps            ';
         'homogen_hildesheimer_15_go_1000_steps  ';
         'homogen_suedstadt_15_go_1000_steps     '];

% Information about scenario
% 0 = hetero, 1 = homo)
scenario = [0, 0, 1, 1];
% Information about location
% 0 = Hildesheimer, 1 = Suedstadt, 2 = Grid
location = [0, 1, 0, 1];

% Number of files
n_files = size(files, 1);
% Burn in phase
burn_in = 250;
```

Listing C.15: Simulation Setup.

C.4.3 Data Import

This Section imports the data and combines it to a overall data set which consists of two scenarios (homogenous and heterogenous) and two locations (Hildesheimer, Südstadt). The artificial simulation data was not considered. For example grid contained only 500 steps as a performance problem.

```
% Import .mat files
for i = 1:n_files

    % Build file names
    file = strtrim(files(i,:));
    file_import = ['01_data/', file, '.mat'];

    % Import
    load(file_import);

    % Add indicators for scenario
    dist(:, end + 1) = scenario(i);
    groups(:, end + 1) = scenario(i);
    params(:, end + 1) = scenario(i);

    % Add indicators about location
    dist(:, end + 1) = location(i);
    groups(:, end + 1) = location(i);
    params(:, end + 1) = location(i);

    % Append data
    if i == 1
        data_dist = dist;
        data_groups = groups;
        data_params = params;
    else
        data_dist = vertcat(data_dist, dist);
        data_groups = vertcat(data_groups, groups);
        data_params = vertcat(data_params, params);
    end

    % Remove old variables
    clear dist groups params

end
```

Listing C.16: Data Import.

C.4.4 Statistic Analysis Distance

```
% Build table
table_dist = table;
table_dist.mean_dist = data_dist(:,2);
table_dist.sd_dist = data_dist(:,3);
table_dist.scenario = data_dist(:,4);
table_dist.location = data_dist(:,5);
% Descriptive statistics total
grpstats(table_dist)
% Deskriptive statistics per group
grpstats(table_dist, [3, 4])
% Variable to plot
vars_dist = ['mean_dist';
             'sd_dist  '];
% Number of variables
n_vars_dist = size(vars_dist, 1);
% Loop over vars and plot
for i = 1:n_vars_dist
    % Current variable
    var_plot = strtrim(vars_dist(i, :));
    % Plot steps
    plot_steps(table_dist, var_plot, burn_in, 1000)
    % Regression model
    LinearModel.fit(table_dist, [var_plot, ' ~ scenario * location'])
end
```

Listing C.17: Statistic Analysis Distance.

Linear regression model in the formula form using Wilkinson notation. Here it corresponds to:

Table C.2: Mean Distance: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	5.8088	0.11781	49.306	0
scenario	-0.99018	0.16661	-5.9431	3.0353e-09
location	13.414	0.16661	80.51	0
scenario:location	-0.6273	0.23562	-2.6623	0.0077913

Table C.3: Mean Variation: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	42.065	0.38704	108.68	0
scenario	-4.0104	0.54736	-7.3268	2.8377e-13
location	48.039	0.54736	87.765	0
scenario:location	-0.0031792	0.77408	-0.0041071	0.99672

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 3.73 R-squared: 0.758, Adjusted R-Squared 0.757 F-statistic vs. constant model: 4.16e+03, p-value = 0

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 12.2 R-squared: 0.795, Adjusted R-Squared 0.795 F-statistic vs. constant model: 5.17e+03, p-value = 0

C.4.5 Statistic Analysis Group Formation

```matlab
% Build table
table_group = table;
table_group.n_veh = data_groups(:,2);
table_group.n_group = data_groups(:,3);
table_group.size_group = data_groups(:,4);
table_group.scenario = data_groups(:,5);
table_group.location = data_groups(:,6);
% Descriptive statistics total
grpstats(table_group)
% Deskriptive statistics per group
grpstats(table_group, [4, 5])
% Variable to plot
vars_group = ['n_veh     ';
              'n_group   ';
              'size_group'];

% Number of variables
n_vars_group = size(vars_group, 1);
% Loop over vars and plot
for i = 1:n_vars_group
    % Current variable
    var_plot = strtrim(vars_group(i, :));
    % Plot steps
    plot_steps(table_group, var_plot, burn_in, 1000)
    % Regression model
    LinearModel.fit(table_group, [var_plot, ' ~ scenario * location'])
end
```

Listing C.18: Statistic Analysis Group Formation.

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 41.5 R-squared: 0.646, Adjusted R-Squared 0.646 F-statistic vs. constant model: 2.43e+03, p-value = 0

Table C.4: Group Formation: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	85.507	1.3126	65.142	0
scenario	-15.523	1.8563	-8.3622	8.3988e-17
location	111.96	1.8563	60.312	0
scenario:location	-2.113	2.6252	-0.80488	0.42094

Table C.5: Group Formation - amount of groups: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	11.307	0.16982	66.582	0
scenario	-1.848	0.24016	-7.6948	1.7739e-14
location	14.331	0.24016	59.672	0
scenario:location	-0.072	0.33964	-0.21199	0.83213

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 5.37 R-squared: 0.643, Adjusted R-Squared 0.643 F-statistic vs. constant model: 2.4e+03, p-value = 0

Estimated Coefficients:

Number of observations: 3968, Error degrees of freedom: 3964 Root Mean Squared Error: 0.458 R-squared: 0.0567, Adjusted R-Squared 0.056 F-statistic vs. constant model: 79.5, p-value = 6.33e-50

C.4.6 Statistic Analysis Parameter

```
% Build table
table_params = table;
table_params.alpha_1 = data_params(:,2);
table_params.alpha_2 = data_params(:,3);
table_params.alpha_3 = data_params(:,4);
table_params.alpha_4 = data_params(:,5);
table_params.alpha_5 = data_params(:,6);
table_params.alpha_6 = data_params(:,7);
table_params.pos_x = data_params(:,8);
table_params.pos_y = data_params(:,9);
table_params.dest_x = data_params(:,10);
table_params.dest_y = data_params(:,11);
table_params.max_speed = data_params(:,12);
table_params.max_accel = data_params(:,13);
table_params.max_decel = data_params(:,14);
table_params.cur_speed = data_params(:,15);
table_params.sum_stand = data_params(:,16);
table_params.mean_stand = data_params(:,17);
table_params.scenario = data_params(:,18);
```

Table C.6: Group Formation - size groups: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	7.5314	0.014543	517.88	0
scenario	-0.17483	0.020582	-8.4944	2.7763e-17
location	0.13929	0.020546	6.7796	1.3837e-11
scenario:location	0.058046	0.029071	1.9967	0.045927

Table C.7: Parameter - αx: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	0.16517	0.00046953	351.77	0
scenario	-0.0005	0.00066401	-0.753	0.4515
location	0.00066667	0.00066401	1.004	0.31544
scenario:location	0.00033333	0.00093906	0.35497	0.72263

Table C.8: Parameter - position x: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	491.29	2.1899	224.35	0
scenario	5.0502	3.0969	1.6307	0.10303
location	267.53	3.0969	86.385	0
scenario:location	5.8899	4.3797	1.3448	0.17877

```
table_params.location = data_params(:,19);
% Descriptive statistics total
grpstats(table_params)
% Deskriptive statistics per group
grpstats(table_params, [17, 18])
% Variable to plot
vars_params = ['alpha_1   ';
               'alpha_2   ';
               'alpha_3   ';
               'alpha_4   ';
               'alpha_5   ';
               'alpha_6   ';
               'pos_x     ';
               'pos_y     ';
               'dest_x    ';
               'dest_y    ';
               'max_speed ';
               'max_accel ';
               'max_decel ';
               'cur_speed ';
               'sum_stand ';
               'mean_stand'];

% Number of variables
n_vars_params = size(vars_params, 1);
% Loop over vars and plot
for i = 1:n_vars_params
    % Current variable
    var_plot = strtrim(vars_params(i, :));
    % Plot steps
    plot_steps(table_params, var_plot, burn_in, 1000)
    % Regression model
    LinearModel.fit(table_params, [var_plot, ' ~ scenario * location'])
end
```

Listing C.19: Statistic Analysis Parameter.

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 0.0148 R-squared: 0.000945, Adjusted R-Squared 0.000195 F-statistic vs. constant model: 1.26, p-value = 0.286

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 69.2 R-squared: 0.793, Adjusted R-Squared 0.792 F-statistic vs. constant model: 5.09e+03, p-value = 0

Estimated Coefficients:

Table C.9: Parameter - position y: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	923.64	3.5119	263	0
scenario	-14.182	4.9666	-2.8555	0.0043191
location	242.69	4.9666	48.864	0
scenario:location	5.6822	7.0238	0.80899	0.41857

Table C.10: Parameter - destination x: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	421.22	2.2824	184.55	0
scenario	13.607	3.2278	4.2155	2.5479e-05
location	302.21	3.2278	93.625	0
scenario:location	-30.874	4.5649	-6.7634	1.5433e-11

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 111 R-squared: 0.551, Adjusted R-Squared 0.55 F-statistic vs. constant model: 1.63e+03, p-value = 0

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 72.2 R-squared: 0.798, Adjusted R-Squared 0.798 F-statistic vs. constant model: 5.28e+03, p-value = 0

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 124 R-squared: 0.346, Adjusted R-Squared 0.345 F-statistic vs. constant model: 703, p-value = 0

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 4.93 R-squared: 0.422, Adjusted R-Squared 0.421 F-statistic vs. constant model: 971, p-value = 0

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 0.363 R-squared: 0.737, Adjusted R-Squared 0.737 F-statistic vs. constant model: 3.74e+03, p-value = 0

Estimated Coefficients:

Table C.11: Parameter - Destination y: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	1032.5	3.9263	262.97	0
scenario	-24.472	5.5526	-4.4073	1.0744e-05
location	183.04	5.5526	32.965	5.1374e-211
scenario:location	-10.5	7.8525	-1.3372	0.18124

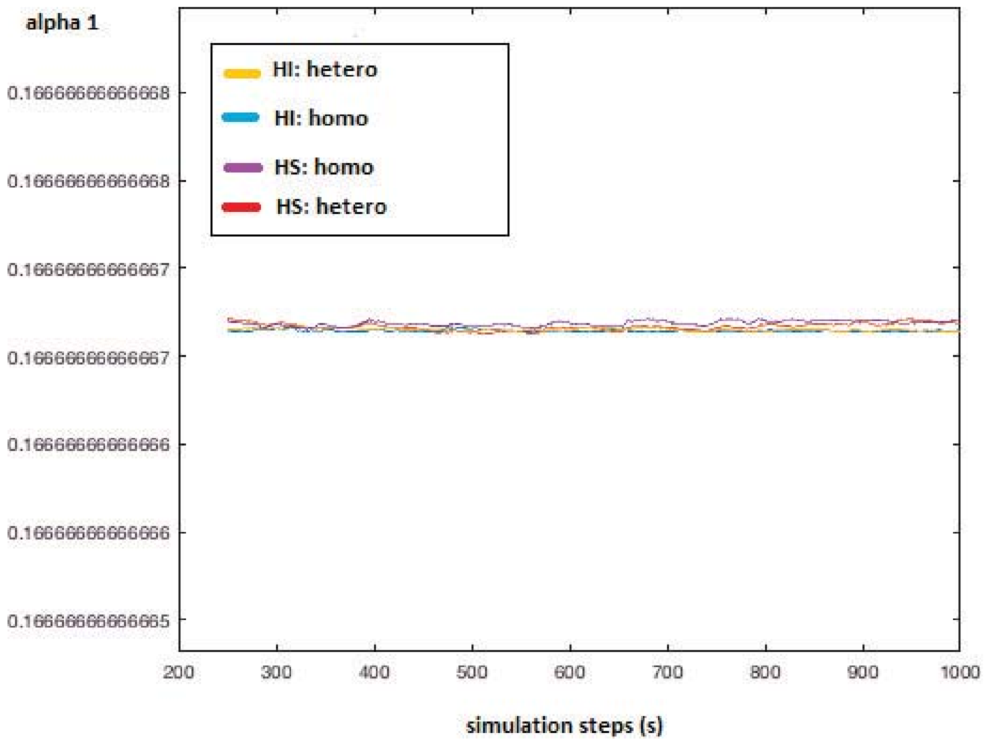


Figure C.8: Alpha 1.

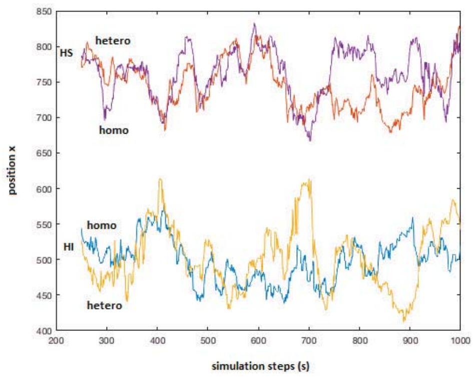


Figure C.9: x Position.

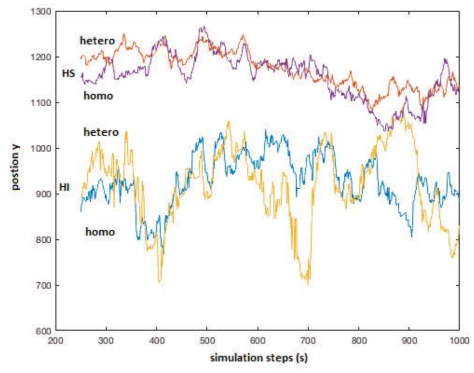


Figure C.10: y Position.

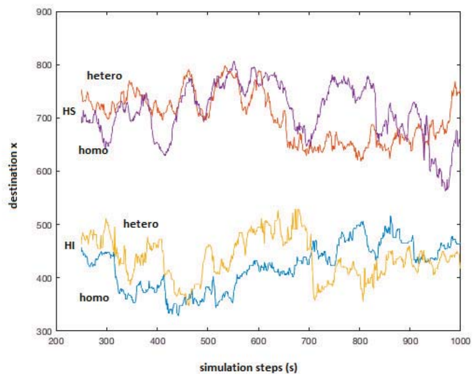


Figure C.11: x Destination.

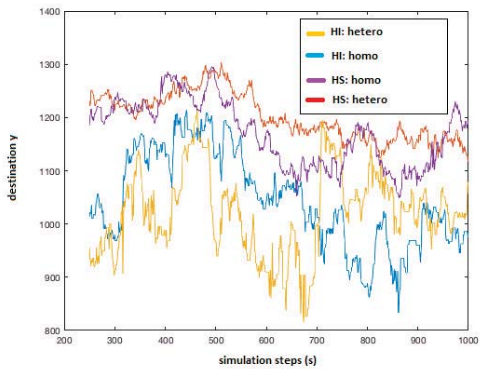


Figure C.12: y Destination.

Table C.12: Parameter - Maximum Speed: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	40.973	0.15603	262.59	0
scenario	8.4274	0.22066	38.192	2.4142e-272
location	0.32573	0.22066	1.4762	0.13998
scenario:location	-0.025734	0.31206	-0.082464	0.93428

Table C.13: Parameter - Maximum Acceleration: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	1.7464	0.011468	152.28	0
scenario	1.2176	0.016219	75.071	0
location	0.023963	0.016219	1.4775	0.13962
scenario:location	-0.0059632	0.022937	-0.25998	0.79489

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 0.492 R-squared: 0.681, Adjusted R-Squared 0.681 F-statistic vs. constant model: 2.85e+03, p-value = 0

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 3.36 R-squared: 0.0399, Adjusted R-Squared 0.0392 F-statistic vs. constant model: 55.4, p-value = 4.5e-35

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 4.56e+04 R-squared: 0.239, Adjusted R-Squared 0.239 F-statistic vs. constant model: 419, p-value = 1.55e-236

Estimated Coefficients:

Number of observations: 4000, Error degrees of freedom: 3996 Root Mean Squared Error: 0.0906 R-squared: 0.00119, Adjusted R-Squared 0.000439 F-statistic vs. constant model: 1.59, p-value = 0.191

Table C.14: Parameter - Maximum Deceleration: Estimated Coefficients Table.

	Estimate	**SE**	**tStat**	**pValue**
(Intercept)	4.4975	0.015544	289.34	0
scenario	1.4305	0.021983	65.072	0
location	0.023933	0.021983	1.0887	0.27635
scenario:location	0.012067	0.031088	0.38816	0.69792

Table C.15: Parameter - Current Speed: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	7.3889	0.1062	69.575	0
scenario	1.4358	0.15019	9.56	1.9932e-21
location	-0.37473	0.15019	-2.495	0.012634
scenario:location	-0.95891	0.2124	-4.5146	6.5252e-06

Table C.16: Parameter - Total Stop Times: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	39012	1441.2	27.069	2.6123e-148
scenario	-8184.6	2038.2	-4.0156	6.0384e-05
location	48455	2038.2	23.773	5.8653e-117
scenario:location	4460.6	2882.5	1.5475	0.12182

Table C.17: Parameter - Average Stop Times: Estimated Coefficients Table.

	Estimate	SE	tStat	pValue
(Intercept)	0.98553	0.002864	344.11	0
scenario	-0.0035114	0.0040503	-0.86694	0.38603
location	0.0046561	0.0040503	1.1496	0.25039
scenario:location	0.0019307	0.005728	0.33707	0.73608

Urban transport is a political and not a technical issue. The technical aspects are very simple. The difficult decisions relate to who is going to benefit from the models adopted.
Enrique Penalosa (1954)

Appendix D

AIMSUN Groups

In order to illustrate the role-based vehicle group formation in form of an adapted contract net protocol to realistic traffic. The listing shows the vehicle class by 'AIMSUN extended' where everything is defined:

```
Vehicle::Vehicle(InfVeh vehTrackedInf,
StaticInfVeh vehTrackedStaticInf,
DataModel* dataModel)
{
    id_ = vehTrackedInf.idVeh;
    current_speed = vehTrackedInf.CurrentSpeed;
    previous_speed = vehTrackedInf.PreviousSpeed;
    type = vehTrackedStaticInf.type;
    pos.UpdateXPosition(vehTrackedInf.xCurrentPos);
    pos.UpdateYPosition(vehTrackedInf.yCurrentPos);
    pos.UpdateZPosition(vehTrackedInf.zCurrentPos);
    section_position = vehTrackedInf.CurrentPos;
    lane_number = vehTrackedInf.numberLane;
    status = vehTrackedInf.report;
    section_id = vehTrackedInf.idSection;
    enrouted = vehTrackedStaticInf.enrouted;
    tracked = vehTrackedStaticInf.tracked;
    equipped = vehTrackedStaticInf.equipped;
    distance2end = vehTrackedInf.distance2End;
    section_id_next = aimsun_->GetNextSection
      (vehTrackedInf.idVeh,vehTrackedInf.idSection);
    centroid_id_orig = vehTrackedStaticInf.centroidOrigin;
    centroid_id_dest = vehTrackedStaticInf.centroidDest;
    section_id_From = AKIInfNetGetIdSectionofOriginCentroidConnector
      (vehTrackedStaticInf.centroidOrigin, 0);
    section_id_To = AKIInfNetGetIdSectionofDestinationCentroidConnector
      (vehTrackedStaticInf.centroidDest, 0);

    comsys_ = new ComSys(vehTrackedInf.idVeh, dataModel, "mobile");

    isLeader_ = false; //for grouping
    isMember_ = true;
}
```

Listing D.1: Description of the vehicle class.

```
void GroupApp::setLeader()
{
    double distance2end = dataModel_->getVehicle(id_)
        ->GetDistance2End();
    int sg_id = this->getSignalGroup();

    for(map<int, Message>::iterator message =
      incomingTLMessage_.begin(); message
      != incomingTLMessage_.end(); message++)
    {
        if (message->second.senderID_ == intersection_id_
          && message->second.protocolMsg == TL)
        {

           for (int i=0; i <
           (signed int) message->second.sg.size(); i++)
          {

             if (sg_id == message->second.sg.at(i).sg_ID)
              {
                  curr_sg_ = message->second.sg.at(i);
                  break;

              }
          }
        }
    }
    double TL_state = curr_sg_.sg_State;
    int destination_centroid = dataModel_->getVehicle(id_)
      ->getCentroidIdDest();
    int next_section = dataModel_->getVehicle(id_)
      ->GetIdSectionNext();

    //set Leader wenn vehicle stops at the red light
    if(destination_centroid == centroid_id_1_
      || destination_centroid == centroid_id_2_ ||
       destination_centroid == centroid_id_3_
        || destination_centroid == centroid_id_4_ ||
       next_section == next_hildes_nord_id_){
      if (distance2end < 3.7 && TL_state == 0)
       {
           dataModel_->getVehicle(id_)->setLeader(true);
       }
    }
}
```

Listing D.2: Description of GroupApp.

```
double GroupApp::calculateOptimalSpeed
(double distance_leader, double distance_stop_line,
 double t_restgreen, double v_max)
{
 double v_min = 30.0;  //30 km/h
 double v_opt = calculateRealOptimalSpeed
 (distance_stop_line + distance_leader, t_restgreen);

 v_opt = v_opt + v_opt/5;

 if (v_opt > v_max){
     return v_max;
  } else if (v_opt < v_min) {
    return v_min;
   }else{
   return v_opt;
   }
 return -1.0;
}
```

Listing D.3: Calculation of Optimal Speed.

The calculation of the real optimal speed is done in the method 'calculate-RealOptimalSpeed'.

```
double GroupApp::calculateRealOptimalSpeed
(double distance_stop_line, double t_restgreen)
{
 return 3.6*(distance_stop_line)/t_restgreen;
}
```

Listing D.4: Calculation of REAL Optimal Speed.

Plan 1 - supportive flow diagram better than code?!

```
if(grouping_->getGroupPlan() == 1) {
  if(!(tl_msg.empty()))
     for(map<int, Message>::iterator message =
         tl_msg.begin(); message != tl_msg.end(); message++)
  {
     if(grouping_->isOnSection
       (message->second.senderID_, curr_sect_id))
     {
```

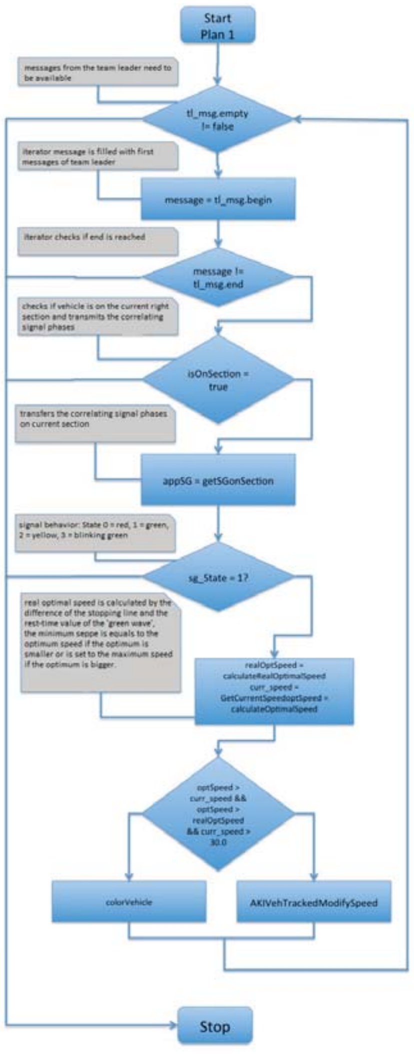

Start
Plan 1
messages from the team leader need to be available
tl_msg.empty != false
iterator message is filled with first messages of team leader
message = tl_msg.begin
iterator checks if end is reached
message != tl_msg.end
checks if vehicle is on the current right section and transmits the correlating signal phases
isOnSection = true
transfers the correlating signal phases on current section
appSG = getSGonSection
signal behavior: State 0 = red, 1 = green, 2 = yellow, 3 = blinking green
sg_State = 1?
real optimal speed is calculated by the difference of the stopping line and the rest-time value of the 'green wave', the minimum seppe is equals to the optimum speed if the optimum is smaller or is set to the maximum speed if the optimum is bigger.
realOptSpeed = calculateRealOptimalSpeed curr_speed = GetCurrentSpeedoptSpeed = calculateOptimalSpeed
optSpeed > curr_speed && optSpeed > realOptSpeed && curr_speed > 30.0
colorVehicle
AKIVehTrackedModifySpeed
Stop

Figure D.1: AIMSUN Leader.

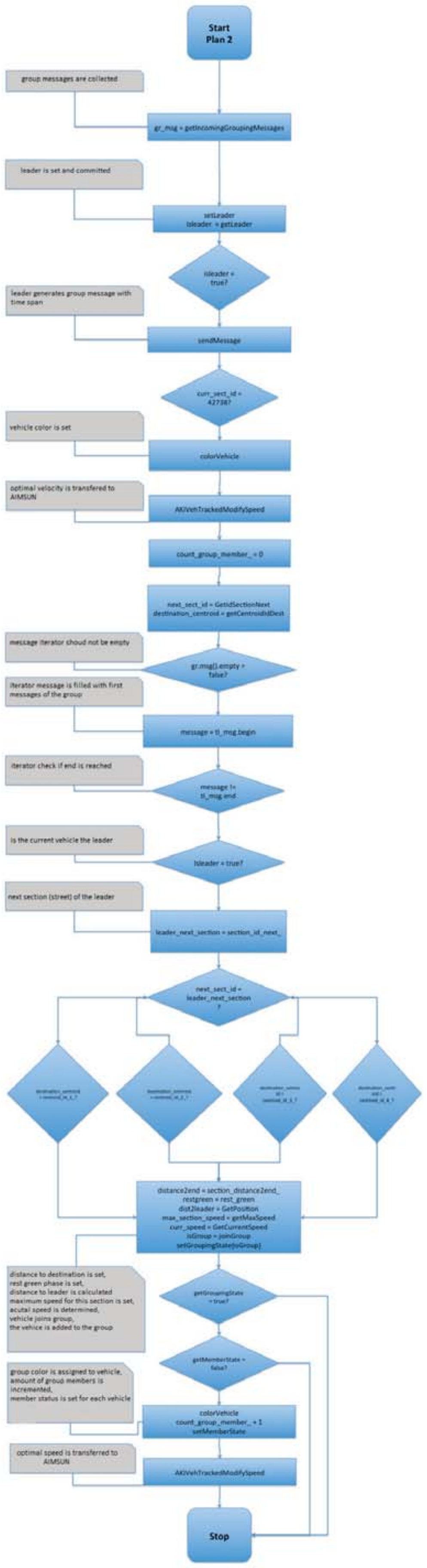

Start
Plan 2
group messages are collected
gr_msg = getIncomingGroupingMessages
leader is set and committed
setLeader
isleader = getLeader
isleader = true?
leader generates group message with time span
sendMessage
curr_sect_id = 42738?
vehicle color is set
colorVehicle
optimal velocity is transfered to AIMSUN
AKIVehTrackedModifySpeed
count_group_member_ = 0
next_sect_id = GetIdSectionNext
destination_centroid = getCentroidIdDest
message iterator choud not be empty
gr.msg().empty = false?
iterator message is filled with first messages of the group
message = tl_msg.begin
iterator check if end is reached
message != tl_msg.end
is the current vehicle the leader
Isleader = true?
next section (street) of the leader
leader_next_section = section_id_next_
next_sect_id = leader_next_section ?
distance2end = section_distance2end_
restgreen = rest_green
dist2leader = GetPosition
max_section_speed = getMaxSpeed
curr_speed = GetCurrentSpeed
isGroup = joinGroup
setGroupingState(isGroup)
distance to destination is set, rest green phase is set, distance to leader is calculated maximum speed for this section is set, acutal speed is determined, vehicle joins group, the vehice is added to the group
getGroupingState = true?
getMemberState = false?
group color is assigned to vehicle, amount of group members is incremented, member status is set for each vehicle
colorVehicle
count_group_member_ + 1
setMemberState
optimal speed is transferred to AIMSUN
AKIVehTrackedModifySpeed
Stop

Figure D.2: AIMSUN Member.

Figure D.3: Flow Diagrams for Grouping by Leader and Member.

```
      signalGroupMessage appSG = grouping_
      ->getSGonSection(curr_sect_id, message->second.sg);
      if(appSG.sg_State == 1){
         double distance2end = dataModel_
         ->getVehicle(id_)->GetDistance2End();
         //calculate the optimal speed for each vehicle
         double optSpeed = grouping_->calculateOptimalSpeed
         (0.0, distance2end, appSG.rest_green, 70.0);
         double realOptSpeed = grouping_->calculateRealOptimalSpeed
         (distance2end, appSG.rest_green);
         double curr_speed = dataModel_
         ->getVehicle(id_)->GetCurrentSpeed();
      if(optSpeed > curr_speed && optSpeed
        > realOptSpeed && curr_speed > 30.0)
           {
            dataModel_->aimsun
             ->colorVehicle(id_, true, 2, grouping_color);
            AKIVehTrackedModifySpeed(id_, optSpeed);
           }
             }
      }
  }
```

Listing D.5: Plan1: Group Leader Plan.

Plan 2:

```
    if(grouping_->getGroupPlan() == 3)
    {

    map<int, Message> gr_msg = dataModel_
    ->getVehicle(id_)
    ->getComSys()->getGroupingApp(
    ->getIncomingGroupingMessages();
    grouping_->setLeader();

    bool isleader = dataModel_
    ->getVehicle(id_)->getLeader();
    if(isleader){
       if(curr_sect_id == 42738)
       {
        grouping_->sendMessage(time);
       }
       dataModel_->aimsun->colorVehicle
             (id_, isleader, 3, grouping_color);
       AKIVehTrackedModifySpeed
             (id_, grouping_->getOptimalSpeed());
             dataModel_->count_group_member_ = 0;
    }
    else{
      int next_sect_id = dataModel_
      ->getVehicle(id_)->GetIdSectionNext();
      int destination_centroid = dataModel_
      ->getVehicle(id_)->getCentroidIdDest();
      if(!(gr_msg.empty()))
        for(map<int, Message>::iterator message =
        gr_msg.begin(); message != gr_msg.end(); message++)
        {
         if(abs(message->second.generationTime_
          - dataModel_->getSimTime()) < 1)
          {
           if (message->second.messageContent_.isleader)
           {
            int leader_next_section =
            message->second.messageContent_.section_id_next_;
            if(next_sect_id == leader_next_section)
            {
             if(destination_centroid == grouping_->centroid_id_1_
             || destination_centroid == grouping_->centroid_id_2_
             || destination_centroid == grouping_->centroid_id_3_
             || destination_centroid == grouping_->centroid_id_4_ )
             {
              double distance2end =
              message->second.messageContent_.section_distance2end_;
              int restgreen = message->second.sg_m.rest_green;
              double dist2leader =
              message->second.messageContent_.position_.GetDistanceFrom
              (dataModel_->getVehicle(id_)->GetPosition());
              double max_section_speed = dataModel_
              ->getRoadSection(curr_sect_id).getMaxSpeed();
              double curr_speed = dataModel_
              ->getVehicle(id_)->GetCurrentSpeed();
              bool isGroup = grouping_->joinGroup
              (dist2leader, message->second.messageContent_,
                  optimalSpeed_, restgreen);
              grouping_->setGroupingState(isGroup);
             }
            }
           }
     break;
     }
    }
```

Listing D.6: Plan 2: Group Member Plan.

```
<route id=  route66   edges=    e1 e2 e3 e4      en   />
```

Attributes

- Id: unique identity
- Edges: set of edges the vehicles is planned to drive. They have to be ordered!

create rou.xml-files
Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE routes[
<!ENTITY vehicleTypes SYSTEM
  "../../general/environmentFiles/vehicleTypes.xml">]>
<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation=
 "http://sumo.sf.net/xsd/routes_file.xsd">

&vehicleTypes;
<route id="route01" edges="e1 e2"/>
<route id="route02" edges="e1 e3"/>
<vehicle depart="1"  id="agent1" route="route01"
type="golf" agenttype="halloAgent" />
<vehicle depart="1"  id="agent2" route="route02"
type="golf" agenttype="halloAgent" />
</routes>
```

Listing D.7: Description of Route File.

Flow A flow generates cars. It is possible to define a certain time window or like in this case a maximum number of cars. Here 1000.
Attributes:

- Id: unique identiy
- Begin: starting point in the simulation in seconds (?). Here the flow starts from the very beginning.
- vehsPerHour: number of cars per hour generated
- number: maximum number to generate. If reached the flow stops
- route: route the cars shall take. One route per flow.
- type: sets the vehicleType to generate. Can be replaced by a distribution.

Optionally it is possible to define a distribution of the vehicle types to be generated.

```
<vType propability=  0.5   />
<vType propability=  0.5   />
```

In every detail a city should reflect that human beings are sacred and that they are equal.
Enrique Penalosa (1954))

Appendix E

ATSim Groups

The following Listing E.1 represents the JAVA class for the vehicle agent written with the FIPA communication standard which basically functions step-based and implements the `Gipps` car-following model and calculates its speed.

```java
import jade.lang.acl.ACLMessage;
import java.io.IOException;
import java.util.Map;
import mas.MasController; //ATSim Architecture Controller
import mas.core.TOAgentState; //State based utiltity
import mas.core.data.VehicleDynamicInfor;
import mas.core.data.VehicleInfor;
import mas.core.data.VehicleStaticInfor;
import mas.util.NameDefine;
import agent.layer.re.behaviours.BehaviourKeepOnMyLane;
import agent.layer.wi.communication.protocols.VehiclePostStepReply;
import agent.layer.wi.communication.protocols.VehiclePreStepReply;
import agent.layer.wi.communication.protocols.NewStateStep;
import agent.memory.ExtendedBelief; //agent belief
public class DemoVehicleAgent extends mas.core.TOAgentAbstract {
 private static final long serialVersionUID = 1L;
private ExtendedBelief memory;
public DemoVehicleAgent(agent.memory.ExtendedBelief memory) {
    super(memory);
    this.memory=memory;
    }
@Override
  protected void setup() {
    super.setup();
  }
@Override
public void postStep() {
  VehiclePostStepReply vps  = new VehiclePostStepReply();
  vps.result=0;
  vps.senderid=this.myid;
  this.agent_state.setPoststepState(vps);
}
@Override
public void preStep() {
  VehiclePreStepReply re = new VehiclePreStepReply();
  re.senderid=this.myid;
  re.result=0;
  this.agent_state.setPrestepState(re);
}
@Override
public void step() {
  Integer pre = this.memory.getNearestPreVehicle();
  double speed=this.followingModel(this.myid, pre);
  double minspeed=this.nextSpeed(this.memory.getDynamicInfor()
  .CurrentSpeed,
  this.memory.getStaticInfor().maxDeceleration,
  this.memory.getStaticInfor().maxDesiredSpeed,
  0, MasController.stepduration);
  speed=Math.max(speed, minspeed);
  NewStateStep vsr=new NewStateStep();
  vsr.newSpeed=speed;
  vsr.newLane=0;
  vsr.idVeh=this.myid;
  this.agent_state.setStepState(vsr);
}
public double followingModel(int follower, int pre) {
  VehicleDynamicInfor followerdinf =
    memory.getDynamicInfOf(follower);
  VehicleStaticInfor followersinf =
    memory.getStaticInfOf(follower);
  double fmaxspeed =
    this.nextSpeed(followerdinf.CurrentSpeed,
    followersinf.maxAcceleration,
    followersinf.maxDesiredSpeed,
    0.0,MasController.stepduration);
  //double fminspeed=this.nextSpeed(followerdinf.CurrentSpeed,
    followersinf.maxDeceleration,
    followersinf.maxDesiredSpeed,
    0.0,MasController.stepduration);
  double sp=fmaxspeed;
  if(follower==pre){
    try {
      throw new Exception();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  if(pre != ExtendedBelief.VEH_NOT_EXIST){
    VehicleDynamicInfor predyn =
      this.memory.getDynamicInfOf(pre);
    VehicleStaticInfor presinf=
      memory.getStaticInfOf(pre);
  //double prenextpos =
  this.nextpos(predyn.CurrentSpeed,
  presinf.maxDeceleration,
  predyn.CurrentPos);
  //double npos =
  this.nextpos(predyn.CurrentSpeed,
  presinf.maxDeceleration,
  predyn.CurrentPos,
  presinf.maxDesiredSpeed);
  //double prenextpos =
    npos;//predyn.CurrentPos +
    this.stopDis(predyn.CurrentSpeed,
    presinf.maxDeceleration);
  ///stopDis is the problem for following model
  //double cspeed = followerdinf.CurrentSpeed;
  //double cpos = followerdinf.CurrentPos;
  double t = MasController.stepduration;
  //double s=prenextpos-cpos-rc.getSecGap(follower,pre);
  //gipps
  double fdcc = followersinf.maxDeceleration;
  double fcpos=followerdinf.CurrentPos;
  double fcspeed=followerdinf.CurrentSpeed;
  double pcpos = predyn.CurrentPos;
  double pcspeed=predyn.CurrentSpeed;
  double pdcc=presinf.maxDeceleration;
  double gippspeed=fdcc*t+Math.sqrt(fdcc*fdcc*t*t-
    fdcc*(2*(pcpos-memory.getSecGap(follower,pre)-fcpos)-
    fcspeed*t-pcspeed*pcspeed/pdcc));
    sp=Math.min(gippspeed,fmaxspeed);
    }
    return sp;
}
private double nextSpeed(double currentspeed,
  double accOrdcc, double maxlimit,
  double minlimit, double time) {
  double newspeed = currentspeed + accOrdcc*time;
  if(newspeed<minlimit)
    return minlimit;
  else if (newspeed > maxlimit){
    return maxlimit;
  }
  return newspeed;
}
}
```

Listing E.1: VehicleAgent Java Class.

This is the grouping algorithm with the Manhattan Distance Function.

```java
import java.util.Vector;
import mas.util.AgentConfig;
import mas.util.DataReader;
import connectorCORBA.VehicleInfoCORBA;
public class AlgorithmsTest {
  private double lamdaspeed;
  private double lamdapos;
  private double lamdadecc;
  private double lamdaacc;
  private Vector<connectorCORBA. VehicleInfoCORBA> data;
  public AlgorithmsTest(String datapath) {
  DataReader reader = new DataReader();
  data = reader.readVehicleData(datapath);
  }
  public void test() {
    for( VehicleInfoCORBA d:data){
      if(d.staticInfo.idVeh==453){
    double acc = d.staticInfo.maxAcceleration;
    double dcc = d.staticInfo.maxDeceleration;
    double dspeed=d.staticInfo.maxDesiredSpeed;
    double mindis = d.staticInfo.minDistanceVeh;
    double cuspeed = d.dynamicInfo.CurrentSpeed;
    double pos = d.dynamicInfo.CurrentPos;
    double[] x = {acc,dcc,dspeed,mindis,cuspeed,pos};
    double[] scalar={1.0,1.0,1.0,1.0,1.0,1.0};
    for( VehicleInfoCORBA d2:data){
      if(d2.staticInfo.idVeh!= d.staticInfo.idVeh){
      double acc2 = d2.staticInfo.maxAcceleration;
      double dcc2 = d2.staticInfo.maxDeceleration;
      double dspeed2=d2.staticInfo.maxDesiredSpeed;
      double mindis2 = d2.staticInfo.minDistanceVeh;
      double cuspeed2 = d2.dynamicInfo.CurrentSpeed;
      double pos2 = d2.dynamicInfo.CurrentPos;
      double diffacc = Math.abs(acc-acc2);
      double diffaccn = diffacc/Math.max(Math.abs(acc), Math.abs(acc2));
        double diffpos = Math.abs(pos-pos2);
        double diffposn = diffpos/Math.max(Math.abs(pos), Math.abs(pos2));
        double diffdspeed = Math.abs(dspeed-dspeed2);
        double diffdspeedn = diffdspeed/Math.max(Math.abs(dspeed),Math.abs(dspeed2));
        double diffcuspeed, diffcuspeedn;
          if(cuspeed==0 && cuspeed2==0) {
            diffcuspeed=0.0;
            diffcuspeedn=0.0;
            }
          else{
            diffcuspeed=Math.abs(cuspeed-cuspeed2);
            diffcuspeedn=diffcuspeed/Math.max(Math.abs(cuspeed),
             Math.abs(cuspeed2));
            }
          double speedf = cuspeed+acc*0.5;
          double speedf2 = cuspeed2+acc2*0.5;
          double diffspeedf = Math.abs(speedf-speedf2);
          double diffspeedfn = diffspeedf/Math.max(Math.abs(speedf),
                Math.abs(speedf2));
          double diffdcc = Math.abs(dcc-dcc2);///Math.max(dcc, dcc2);
          double diffdccn =diffdcc/Math.max(Math.abs(dcc), Math.abs(dcc2));
          double diff = diffdcc+diffdspeed+diffacc+diffcuspeed;
          double diffn=diffdccn+diffdspeedn+diffaccn+diffcuspeedn;
          System.out.println(""+d2.staticInfo.idVeh+" "+diffpos+" "+diffposn);
          }
        }
      }
    }
  }
 public void test2() {
  for( VehicleInfoCORBA d:data){
    if(d.staticInfo.idVeh==453){
    double acc = d.staticInfo.maxAcceleration;
    double dcc = d.staticInfo.maxDeceleration;
    double dspeed=d.staticInfo.maxDesiredSpeed;
    double mindis = d.staticInfo.minDistanceVeh;
    double cuspeed = d.dynamicInfo.CurrentSpeed;
    double pos = d.dynamicInfo.CurrentPos;
    double[] x = {acc,dcc,dspeed,mindis,cuspeed,pos};
    double[] scalar={1.0,1.0,1.0,1.0,1.0,1.0};
      double[] allacc = new double[data.size()];
      double[] alldcc = new double[data.size()];
      double[] allcuspeed = new double[data.size()];
      double[] alldspeed = new double[data.size()];
      double[] allidveh = new double[data.size()];
      double[] allpos = new double[data.size()];
      int i=0;
      allacc[i]=acc;
      alldcc[i]=dcc;
      alldspeed[i]=dspeed;
      allcuspeed[i]=cuspeed;
      allidveh[i]=d.staticInfo.idVeh;
      allpos[i]=pos;
      for( VehicleInfoCORBA d2:data){
        if(d2.staticInfo.idVeh!= d.staticInfo.idVeh){
          i++;
          allacc[i]=d2.staticInfo.maxAcceleration;
          alldcc[i]=d2.staticInfo.maxDeceleration;
          alldspeed[i]=d2.staticInfo.maxDesiredSpeed;
          allcuspeed[i]=d2.dynamicInfo.CurrentSpeed;
          allidveh[i]=d2.staticInfo.idVeh;
          allpos[i]=d2.dynamicInfo.CurrentPos;}}
      double[] adiffacc=new double[data.size()];
      double[] adiffdcc=new double[data.size()];
      double[] adiffdspeed=new double[data.size()];
      double[] adiffcuspeed=new double[data.size()];
      double[] adiffpos=new double[data.size()];
      for(int j=0;j<data.size();j++){
        adiffacc[j]=Math.abs(allacc[0]-allacc[j])/AgentConfig.stddeviationacc;
        adiffdcc[j]=Math.abs(alldcc[0]-alldcc[j])/AgentConfig.stddeviationdcc;
        adiffdspeed[j]=Math.abs(alldspeed[0]-alldspeed[j])/AgentConfig.stddeviationdspeed;
        adiffcuspeed[j]=Math.abs(allcuspeed[0]-allcuspeed[j]);
        adiffpos[j]=Math.abs(allpos[0]-allpos[j]);}
      double[] result = new double[data.size()];
      for(int j=0;j<data.size();j++){
        result[j]=adiffacc[j]+adiffdcc[j]+adiffdspeed[j];}
        this.output(result, allidveh, "result");
      }
    }
  }
  /**
   * @param allacc
   */
  private void output(double[] x, double[] allidveh, String name) {
    System.out.println(name);
    for(int i = 0; i< x.length; i++){
      System.out.println(""+allidveh[i]+" "+x[i]);}}
  /**
   * @param allacc
   * @return
   */
  private double[] standardize(double[] x) {
    double minx=x[0];
    double maxx = x[0];
    for(int i = 1; i< x.length;i++){
      if(minx>x[i])   {
        minx = x[i];}
      if(maxx < x[i]){
        maxx=x[i]; }
    }
    for(int i = 0; i < x.length;i++){
      x[i]=(x[i]-minx)/(maxx-minx);}
    return x;
  }
  private double manhattanDis(double[] x, double[] y, double[] scalar) {
    double diffn = 0.0;
    for(int i=0; i< x.length;i++){
      double diff=Math.abs(x[i]-y[i]);
      diffn = scalar[i]*diff/Math.max(Math.abs(x[i]),Math.abs(y[i]));}
    return diffn;
  }
  private double manhattanDis(mas.core.data.VehicleDynamicInfor dinfor, mas.core.data.VehicleStaticInfor
      sinfor, mas.core.data.VehicleDynamicInfor indinfor, mas.core.data.VehicleStaticInfor insinfor) {
    lamdaacc=0.0;
    lamdadecc=0.0;
    lamdapos=1.0;
    lamdaspeed=0.0;
    double differ =
      lamdaacc*Math.abs(sinfor.maxAcceleration-insinfor.maxAcceleration)+
      lamdadecc*Math.abs(sinfor.maxDeceleration-insinfor.maxDeceleration)+
      lamdapos*Math.abs(dinfor.CurrentPos-indinfor.CurrentPos)+
      lamdaspeed*Math.abs(dinfor.CurrentSpeed-indinfor.CurrentSpeed);
    //norm
    double v = sinfor.maxAcceleration+insinfor.maxAcceleration+
    sinfor.maxDeceleration+insinfor.maxDeceleration+
    dinfor.CurrentPos+indinfor.CurrentPos+
    dinfor.CurrentSpeed+indinfor.CurrentSpeed;
    double endValue = differ/v;
    return endValue;
  }
  public static void main(String[] args)
  {
  new AlgorithmsTest("data/cardata/cardata1444.csv").test2();
  }
}
```

Listing E.2: Decentralized Hierachical Group Formation.

Bibliography

[1] Spring Symposium AAAI. Agents with adjustable autonomy. Technical report, Stanford University, California, USA, 1999. (cited on p. 70)

[2] National Highway Traffic Safety Administration. Automated vehicles, 2017. (cited on pp. 2 and 3)

[3] Adrian K. Agogino and Kagan Tumer. Analyzing and visualizing multi-agent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17(2):320–338, 2008. (cited on p. 179)

[4] K. I Ahmed. *Modeling drivers' acceleration and lane changing behavior.* PhD thesis, Massachusetts Institute of Technology, 1999. (cited on p. 32)

[5] Aris Alissandrakis, Michael Biel, Ferdinando Cicalese, Kerstin Dautenhahn, Gernot Falkner, Martin Giurfa, Barbara Hammer, Gunther Heidemann, Wolfgang Kinzel, Reimer Kühn, David E. J. Linden, Randolf Menzel, Wolfram Menzel, et al. *Adaptivity and Learning - An Interdisciplinary Debate.* Springer, 2003. (cited on p. 71)

[6] Robert John Allan. Survey of agent based modelling and simulation tools. Technical report, Science and Technology Facility Council, 2010. (cited on p. 329)

[7] Eduardo Alonso and Esther Mondragón. Agency, learning and animal-based reinforcement learning. In M. Nickles, M. Rovatsos, and G. Weiß, editors, *Agents and Computational Autonomy: Potential, Risks, and Solutions*, Lecture Notes in Computer Science, pages 1–6. Springer Berlin Heidelberg, 2004. (cited on p. 70)

[8] Malte Aschermann, Philipp Kraus, and Jörg P. Müller. Lightjason: A bdi framework inspired by jason. IFI Technical Report Series Ifi-16-04, Department of Informatics, Clausthal University of Technology, 2016. (cited on p. 316)

[9] Tsz-Chiu Au, Neda Shahidi, and Peter Stone. Enforcing liveness in autonomous traffic management. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, August 2011. (cited on pp. 76 and 106)

[10] Robert Axelrod. An evolutionary approach to norms. *American political science review*, 80:1095–1111, 1986. (cited on p. 83)

[11] Robert M. Axelrod. *The complexity of cooperation: Agent-based models of competition and collaboration.* Princeton University Press, 1997. (cited on p. 83)

[12] Jonathan Baber, Julian Kolodko, Tony Noël, Michael Parent, and Ljubo Vlacic. Cooperative autonomous driving: intelligent vehicles sharing city roads. *Robotics & Automation Magazine, IEEE*, 12(1):44–49, 2005. (cited on pp. 33, 70, and 157)

[13] Reinhold Baier, Ulrich Brannolte, Werner Brilon, Lothar Dunker, Gert Hartkopf, Hartmut Keller, Gerd Kellermann, Uwe Köhler, Reinhart Kühne, Reinhold Maier, Michael Rohloff, Werner Schnabel, Wolfgang Wirth, and Heinz Zackor. Handbuch für die Bemessung von Straßenverkehrsanlagen (hbs). Technical report, Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), 2001. Ausgabe 2001, Fassung 2009. (cited on p. 123)

[14] Gabriel Balan and Sean Luke. History-based traffic control. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems AAMAS-2006*, pages 616–621. ACM Press, 2006. (cited on pp. 75 and 106)

[15] M. Balmer, K. Meister, M. Rieser, K. Nagel, Kay W. Axhausen, Kay W. Axhausen, and Kay W. Axhausen. *Agent-based simulation of travel demand: Structure and computational performance of MATSim-T.* ETH, Eidgenössische Technische Hochschule Zürich, IVT Institut für Verkehrsplanung und Transportsysteme, 2008. (cited on p. 35)

[16] Michael Balmer, Nurhan Cetin, Kai Nagel, and Bryan Raney. Towards truly agent-based traffic and mobility simulations. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, pages 60–67, Washington, DC, USA, 2004. IEEE Computer Society. (cited on pp. 35 and 142)

[17] M. Bando, K. Hasebe, K. Nakanishi, and A. Nakayama. Analysis of optimal velocity model with explicit delay. *Physical Review E*, 58:5429–5435, 1998. (cited on p. 325)

[18] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama. Dynamical model of traffic congestion and numerical simulation. *Physical Review E*, 51:1035–1042, 1995. (cited on pp. xvii, 325, and 326)

[19] K. Suzanne Barber and Jisun Park. Agent belief autonomy in open multiagen systems. In M. Nickles, M. Rovatsos, and G. Weiss, editors, *Agents and Computational Autonomy: Potential, Risks, and Solutions*, volume AUTONOMY 2003 of *LNAI*, pages 7–16. Springer, Berlin Heidelberg, 2004. (cited on p. 70)

[20] J. Barceló, E. Codina, J. Casas, J. L. Ferrer, and D. García. Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. *Journal of Intelligent and Robotic Systems*, 2–3:173–203, 2005. (cited on pp. 131, 132, and 134)

[21] Jaume Barceló. *Fundamentals of Traffic Simulation*, volume 145 of *International Series in Operations Research & Management Science*. Springer, 2010. (cited on pp. 28, 31, 35, 105, and 330)

[22] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *In Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 567–574, 2011. (cited on pp. 80 and 131)

[23] L. Baskar, B. De Schutter, J. Hellendoorn, and Z. Papp. Traffic control and intelligent vehicle highway systems: A survey. *IET Intelligent Transport Systems*, 5(1):38–52, 2011. (cited on p. 64)

[24] R. Bauer, H.-J. Bullinger, M. Carbon, B. Ehlers, J. Fleig, G. Gidion, G. Kiemann, R. Schneider, K. Schock, and A. Weller. *Effizientes Informationsmanagement in dezentralen Organisationsstrukturen*. Springer, 1999. (cited on pp. 152 and 153)

[25] Ana Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10:131–164, 2005. (cited on pp. 76 and 105)

[26] Ana L. C. Bazzan, Denise de Oliveira, and Bruno C. da Silva. Learning in groups of traffic signals. *Engineering Applications of Artificial Intelligence*, 23:560–568, 2010. (cited on pp. 106, 132, and 142)

[27] Ana L. C. Bazzan, M. B. do Amarante, and F. B. da Costa. Management of demand and routing in autonomous personal transportation. *Journal of Intelligent Transportation Systems*, 16(1):1–11, 2012. (cited on pp. 68 and 106)

[28] Ana L. C. Bazzan and Franziska Klügl. Re-routing agents in an abstract traffic scenario. In *SBIA*, pages 63–72, 2008. (cited on p. 79)

[29] Ana L. C. Bazzan and Franziska Klügl. A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, FirstView:1–29, 9 2013. (cited on pp. 57, 58, 62, 75, 79, 105, and 140)

[30] Ana L. C. Bazzan, Denise Oliveira, Franziska Klügl, and Kai Nagel. To adapt or not to adapt consequences of adapting driver and traffic light agents. In Karl Tuyls, Ann Nowe, Zahia Guessoum, and Daniel Kudenko, editors, *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, volume 4865 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2008. (cited on p. 79)

[31] Tristan Behrens. *Towards Building Blocks for Agent-Oriented Programming - Standardizing Interpreters, Environments and Tools*. PhD thesis, Clausthal University of Technology, Faculty of Mathematics, Computer Science and Mechanical Engineering, February 2012. (cited on pp. xviii, 52, 53, 139, 148, 347, and 348)

[32] Tristan Behrens, Michael Köster, Federico Schlesinger, Jürgen Dix, and Jomi F. Hübner. The multi-agent programming contest 2011: A resume.

In Louise Dennis, Olivier Boissier, and RafaelH. Bordini, editors, *Programming Multi-Agent Systems*, volume 7217 of *Lecture Notes in Computer Science*, pages 155–172. Springer Berlin Heidelberg, 2012. (cited on p. 52)

[33] Meredith R. Belbin. *Team Roles at Work*. Butterworth-Heinemann, 2010. (cited on p. 16)

[34] R. M. Belbin. *Management Teams*. Routledge, 2, revised edition, 2012. (cited on p. 16)

[35] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE - a FIPA-compliant agent framework. In *Proc. of the Practical Applications of Intelligent Agents*, 1999. (cited on p. 131)

[36] Fabio L. Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. Wiley, April 2007. (cited on pp. 51 and 334)

[37] Carl Bergenhem, Qihui Huang, Ahmed Benmimoun, and Tom Robinson. Challenges of platooning on public motorways. In *17th world congress on intelligent transport systems*, pages 1–12, 2010. (cited on p. 82)

[38] Carl Bergenhem, Henrik Pettersson, Erik Coelingh, Cristofer Englund, Steven Shladover, and Sadayuki Tsugawa. Overview of platooning systems. In *ITS World Congress*, Vienna, Austria, 22-26 October 2012. (cited on pp. 3, 82, and 234)

[39] A. J. Bermejo, J. Villadangos, J. J. Astrain, and A. Cordoba. Ontology based road traffic management. In Giancarlo Fortino, Costin Badica, Michele Malgeri, and Rainer Unland, editors, *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 103–108. Springer Berlin Heidelberg, 2013. (cited on p. 49)

[40] Philippe Bernoux. *La sociologie des organisations: Initiation theorique suivie de douze cas pratiques*. Seuil, 1985. (cited on p. 50)

[41] Michele Bertoncello and Dominik Wee. Ten ways autonomous driving could redefine the automotive world, 2015. (cited on p. 2)

[42] Holger Billhardt, Marin Lujak, Vicente Snchez-Brunete, Alberto Fernndez, and Sascha Ossowski. Dynamic coordination of ambulances for emergency medical assistance services. *Knowledge-Based Systems*, 70(Supplement C):268 – 280, 2014. (cited on p. 315)

[43] Transportation Research Board. Highway capacity manual, 3rd edition, transportation research board 1994, 1994. (cited on pp. xv, 25, and 61)

[44] Florence Boillot, Sophie Midenet, and Jean-Claude Pierrelee. The real-time urban traffic control system cronos: Algorithm and experiments. *Transportation Research Part C: Emerging Technologies*, 14:18–38, 2006. (cited on pp. 63 and 106)

[45] Olivier Boissier. Multi-agent oriented programming - organisation-oriented programming. Website, October 2013. (cited on pp. 38, 50, 141, and 169)

[46] R. Peter Bonasso, David Kortenkamp, David P. Miller, and Marc Slack. Experiences with an architecture for intelligent, reactive agents. *JOURNAL OF EXPERIMENTAL AND THEORETICAL ARTIFICIAL INTELLIGENCE*, 9:237–256, 1995. (cited on p. 43)

[47] Denise A Bonebright. 40 years of storming: a historical review of tuckman's model of small group development. *Human Resource Development International*, 13:111–120, 2010. (cited on pp. 19 and 20)

[48] Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and A El Fallah Seghrouchni. *Multi-Agent Programming*. Springer, 2005. (cited on pp. xv, 44, 52, and 53)

[49] Rafael H. Bordini, Jomi Fred Hbner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley, 2007. (cited on pp. 52, 139, 230, 336, and 376)

[50] Andrei Borshchev. *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*. AnyLogic North America, June 2013. (cited on p. 332)

[51] David Boyce and Huw Williams. *Forecasting Urban Travel: Past, Present and Future*. Edward Elgar Publishing, 2015. (cited on p. 24)

[52] Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355, 1988. (cited on p. 43)

[53] Rodney A. Brooks. A robot that walks; emergent behaviors from a carefully evolved network. *Neural computation*, 1(2):253–262, 1989. (cited on p. 41)

[54] Nicolas Brusson. Wie sich unser Verhältnis zum Auto verändern wird, 2017. (cited on p. 1)

[55] Matt Burgess. Autonomous 'milk floats' are now delivering ocado shopping in london, 2017. (cited on pp. 2 and 3)

[56] Cándido Caballero-Gil, Pino Caballero-Gil, and Jezabel Molina-Gil. Group formation through cooperating node in vanets. In *International Conference on Cooperative Design, Visualization and Engineering*, pages 105–108. Springer, 2010. (cited on p. 82)

[57] Gordon D. B. Cameron and Gordon I. D. Duncan. Paramics: Parallel microscopic simulation of road traffic. *The Journal of Supercomputing*, 10:25–53, 1996. (cited on pp. 132 and 134)

[58] Peter Campbell. Oxbotica unlocks the potential of driverless cars, 2017. (cited on p. 1)

[59] Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. *ICMAS*, 95:41–48, 1995. (cited on p. 170)

[60] Cristiano Castelfranchi. Founding agents 'autonomy' on dependence theory. In *ECAI*, volume 1, pages 353–357, 2000. (cited on p. 70)

[61] D. Charypar, K. Nagel, and K. W. Axhausen. An event-driven queue-based microsimulation of traffic flow. In *Transportation Research Record*, pages 35–40. 2003, 2007. (cited on pp. 67 and 106)

[62] Bo Chen and H.H. Cheng. A review of the applications of agent technology in traffic and transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 11(2):485–497, 2010. (cited on pp. 58 and 140)

[63] Amit K. Chopra, Alexander Artikis, Jamal Bentahar, Marco Colombetti, Frank Dignum, Nicoletta Fornara, Andrew J. I. Jones, Munindar P. Singh, and Pinar Yolum. Research directions in agent communication. *ACM Transactions on Intelligent Systems and Technology*, 4(2):1–23, April 2013. Article 20. (cited on p. 49)

[64] Viet Hung Chu. Verfahren der dynamischen Gruppenbildung zur Koordination autonomer Fahrzeuge im Straßenverkehr. Master's thesis, Technische Universität Clausthal, May 2011. (cited on pp. xvi, 104, 131, 132, 133, 136, 157, 162, 163, 164, 166, 167, 168, 207, and 242)

[65] Viet Hung Chu, Jana Görmer, and Jörg P. Müller. ATSim: Combining AIMSUM and jade for agent-based traffic simulation. In *14th Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*. Springer-Verlag, 2011. (cited on pp. 131, 132, 134, 136, 137, and 335)

[66] Michael Clark, Oliver Ernhofer, Bernard Frayne, Bernhard Friedrich, David Lawrence, Tom McLean, Joachim Mertz, John Parker, and Colin Toomey. Telematics applications in Bavaria, Scotland and others project number tr1054 deliverable number 9.3. Technical report, Cordis Europa, 1998. (cited on pp. xv and 60)

[67] Erik Coelingh. Sartre project completes first successful on-road demo of multiple vehicle platooning. In *Green Car Congress.*, 2012. (cited on p. 4)

[68] Philip R. Cohen and Hector J. Levesque. Teamwork. *Special Issue on Cognitive Science and AI*, 25:487–512, 1991. (cited on pp. 80, 86, and 131)

[69] D. J. Collins, R. I. Grigorchuk, P. F. Kurchanov, and H. Zieschang. *Combinatorial Group Theory and Applications to Geometry.* Springer, 2. printing edition, 1998. (cited on p. 183)

[70] Marco Colombetti, Nicoletta Fornara, and Mario Verdicchio. A social approach to communication in multiagent systems. In Joo Leite, Andrea Omicini, Leon Sterling, and Paolo Torroni, editors, *Declarative Agent Languages and Technologies*, volume 2990 of *Lecture Notes in Computer Science*, pages 191–220. Springer Berlin Heidelberg, 2004. (cited on p. 49)

[71] A. Consoli, J. Tweedale, and L. Jain. An architecture for agent coordination and cooperation. In *Proceedings of KES 2007 11th International Conference on Knowledge-Based Intelligent Engineering Systems*, pages 934–940. Springer, 2007. (cited on pp. 80 and 131)

[72] AdaptIVe Consortium. Adaptive: Automated driving, 2017. (cited on p. 2)

[73] J. Cuena, J. Hernàndez, and M. Molina. An intelligent model for road traffic management in the motorway network around barcelona. In Nobuyoshi Terashima and Edward Altman, editors, *Advanced IT Tools*, IFIP The International Federation for Information Processing, pages 173–180. Springer US, 1996. (cited on pp. 63 and 105)

[74] Denise de Oliveira, Ana L. C. Bazzan, and Victor R. Lesser. Using cooperative mediation to coordinate traffic lights: a case study. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *AAMAS*, pages 463–470. ACM, 2005. (cited on pp. 77 and 105)

[75] Denise de Oliveira, Paulo R. Ferreira Jr., and Ana L. C. Bazzan. A swarm based approach for task allocation in dynamic agents organizations. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 3 of *AAMAS '04*, pages 1252–1253. IEEE Computer Society, 2004. (cited on pp. 78 and 105)

[76] Lucas Barcelos de Oliveira and Eduardo Camponogara. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(1):120 – 139, 2010. Information/Communication Technologies and Travel Behaviour/ Agents in Traffic and Transportation. (cited on pp. 69, 78, and 106)

[77] A. de Palma, M. Ben-Akiva, D. Brownstone, C. Holt, T. Magnac, D. McFadden, P. Moffatt, N. Picard, K. Train, P. Wakker, and J. Walker. Risk, uncertainty and discrete choice models. In *Marketing Letters*. 19, 2008. (cited on pp. 69 and 105)

[78] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst MA 01003-4610, May 1995. (cited on p. 93)

[79] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Eleventh National Conference on Artificial Intelligence*, pages 217–224, Washington, July 1993. (cited on p. 93)

[80] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. UMass Computer Science Technical Report 9414, Department of Computer Science, University of Massachusetts, Amherst, MA 01003, DECKER@CS.UMASS.EDU, August 1995. (cited on pp. 93 and 94)

[81] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press, pages 73–80, San Francisco, June 1995. (cited on pp. 93 and 94)

[82] Sophie L. Dennisen and Jörg P. Müller. Agent-based voting architecture for traffic applications. In Jörg P. Müller, Wolf Ketter, Gal A. Kaminka, Gerd Wagner, and Nils Bulling, editors, *Multiagent System Technologies*

- 13th German Conference, MATES 2015, Cottbus, Germany, September 28-30, 2015, Revised Selected Papers, volume 9433 of *Lecture Notes in Computer Science*, pages 200–217. Springer, 2015. (cited on p. 315)

[83] C. Desjardins, J. Laumônier, and B. Chaib-draa. Learning agents for collaborative driving. In Ana L. C. Bazzan and Franziska Klügl, editors, *Multi-Agent Systems for Traffic and Transportation*, pages 240–260. IGI Global, 2009. (cited on pp. 79 and 105)

[84] Morton Deutsch. A theory of co-operation and competition. *Human Relations*, 2:129–152, 1949. (cited on p. 16)

[85] Christina Diakaki, Markos Papageorgiou, and Kostas Aboudolas. A multi-variable regulator approach to traffic-responsive network-wide signal control. control engineering practice 10, 2002. (cited on pp. 62, 78, and 106)

[86] M. Bernardine Dias and Anthony Stentz. A free market architecture for distributed control of a multirobot system. In *In 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, July 2000. (cited on p. 310)

[87] Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge. A formal specification of dmars. In *Intelligent Agents IV Agent Theories, Architectures, and Languages.* Springer, 1998. (cited on p. 44)

[88] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. On cooperation in multi-agent systems, 1997. (cited on pp. 80 and 131)

[89] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, pages 530–537, Washington, DC, USA, 2004. IEEE Computer Society. (cited on pp. 72, 81, 103, 106, 204, 296, 315, and 316)

[90] Kurt Dresner and Peter Stone. Multiagent traffic management: An improved intersection control mechanism. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, July 2005. ACM Press. (cited on p. 81)

[91] Kurt Dresner and Peter Stone. Multiagent traffic management: Opportunities for multiagent learning. In K. Tuyls et al., editor, *LAMAS 2005*, volume 3898 of *Lecture Notes in Artificial Intelligence*, pages 129–138. Springer Verlag, Berlin, 2006. (cited on pp. 81 and 106)

[92] Kurt Dresner and Peter Stone. Sharing the road: Autonomous vehicles meet human drivers. In *The 20th International Joint Conference on Artificial Intelligence*, pages 263–68, 01 2007. (cited on p. 72)

[93] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31:591–656, March 2008. (cited on p. 81)

[94] E. H. Durfee and V. R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 875–883, Milan, Italy, 1987. (cited on p. 93)

[95] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1:63–83, 1989. (cited on p. 45)

[96] J. Ehmke, S. Meisel, and D. Mattfeld. Floating car data based analysis of urban travel times for the provision of traffic quality. In *Proceedings of the International Workshop on Traffic Data Collection & its Standardization*, Barcelona, Spain, 2008. (cited on p. 282)

[97] Christian Eichhorn. Vergleich agentenorientierter Simulationsplattformen und Evaluation ihrer Verwendbarkeit für Anwendungen des kooperativen Verkehrsmanagements. Wirtschaftsinformatik, M. Sc., Technische Universität Clausthal, Institut für Informatik Abteilung Wirtschaftsinformatik Julius-Albert-Str. 4, Oktober 2012. (cited on pp. xvii, 330, 331, and 346)

[98] Marc Esteva, Bruno Rosell, Juan A. Rodríguez-Aguilar, and Josep Lluís Arcos. Ameli: An agent-based middleware for electronic institutions. In Carles Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 236–243, 2004. (cited on p. 50)

[99] Wolfgang Fastenmeier. Mehr Sicherheit durch autonomes Fahren? *Zeitschrift des Berufsverbandes Psychologinnen und Psychologen e.V.*, 2017. (cited on p. 1)

[100] Jacques Ferber. Simulating with reactive agents. *Many Agent Simulation and Artificial Life*, 36:8–28, 1994. (cited on p. 41)

[101] Jacques Ferber, Olivier Gutknecht, and Fabien Michel. From agents to organizations: an organizational view of multi-agent systems. In *Agent-Oriented Software Engineering IV*, pages 214–230. Springer, 2004. (cited on p. 50)

[102] Innes A Ferguson. *Touring Machines: An architecture for dynamic, rational, mobile agents.* PhD thesis, University of Cambridge, November 1992. (cited on p. 43)

[103] N.V. Findler and G.D. Elder. Multi-agent coordination and cooperation in a distributed dynamic environment with limited resources. *Artificial Intelligence in Engineering*, 9:229–238, 1995. (cited on pp. 80 and 131)

[104] Jelena Fiosina and Maksims Fiosins. Cooperative regression-based forecasting in distributed traffic networks. In Qurban A. Memon, editor, *Distributed Network Intelligence, Security and Applications*, chapter 1, pages 3–37. CRC Press, Taylor and Francis Group, 2013. (cited on pp. 64 and 220)

[105] Jelena Fiosina, Maksims Fiosins, and Jörg P. Müller. Decentralised cooperative agent-based clustering in intelligent traffic clouds. In Matthias

Klusch, Marcin Paprzycki, and Matthias Thimm, editors, *Multiagent System Technologies – Proceedings of the 11th German Conference on Multiagent System Technologies*, volume 8076, pages 59–72. Springer, 2013. (cited on pp. 64 and 220)

[106] M. Fiosins, J. Fiosina, J. P. Müller, and J. Görmer. Agent-based integrated decision making for autonomous vehicles in urban traffic. In *9th International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 173–178. Springer-Verlag, 2011. (cited on pp. 110, 137, 153, 157, 220, and 278)

[107] Maksims Fiosins, Jelena Fiosina, Jörg P. Müller, and Jana Görmer. Reconciling strategic and tactical decision making in agent-oriented simulation of vehicles in urban traffic. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 144–151. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011. (cited on pp. 153, 157, 220, and 278)

[108] Maksims Fiosins, Bernhard Friedrich, Jana Görmer, Dirk Mattfeld, Jörg P. Müller, and Hugues Tchouankem. Autonomic routing and grouping with vehicular communication for (de-)centralized urban traffic management. In *16th International IEEE Conference on Intelligent Transportation Systems*. IEEE, October 2013. (cited on pp. 120, 128, and 233)

[109] Maksims Fiosins, Bernhard Friedrich, Jana Görmer, Dirk Mattfeld, Jörg P. Müller, and Hugues Tchouankem. Autonomic routing and grouping with vehicular communication for (de-) centralized urban traffic management. Website, 2014. (cited on pp. xvii, 137, 218, 220, and 283)

[110] Maksims Fiosins, Bernhard Friedrich, Jana Görmer, Dirk Mattfeld, Jörg P. Müller, and Hugues Tchouankem. *Autonomic Routing and Grouping with Vehicular Communication for (De-) centralized Urban Traffic Management*, chapter A Multiagent Approach to Modeling Autonomic Road Transport Support Systems, pages 67–85. Springer International Publishing, Cham, 2016. (cited on pp. xv, xvii, 5, 162, 199, 220, 225, 232, 234, 238, 267, 280, and 299)

[111] Maksims Fiosins, Jana Görmer, and Jan F. Ehmke. Decentralized decision support and coordination of autonomous vehicles based on online data mining techniques. *24th European Conference on Operations Research*, EURO 2010:183ff., 2010. (cited on pp. 157 and 220)

[112] Maksims Fiosins, Jörg P. Müller, and Michaela Huhn. A norm-based probabilistic decision-making model for autonomic traffic networks. In Juan M. Corchado, Javier Bajo, Jaroslaw Kozlak, Pawel Pawlewski, Jose M. Molina, Vicente Julian, Ricardo Azambuja Silveira, Rainer Unland, and Sylvain Giroux, editors, *Highlights on Practical Applications of Agents and Multi-Agent Systems*, volume 365 of *Communications in Computer and Information Science*, pages 49–60. Springer, 2013. (cited on pp. 153 and 220)

[113] Klaus Fischer, Brahim Chaib-Draa, Jörg P. Müller, Marcus Pischel, and Christian Gerber. A simulation approach based on negotiation and cooperation between agents: a case study. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 29(4):531–545, 1999. (cited on pp. 197 and 198)

[114] Michael Fisher. A survey of concurrent metatem - the language and its applications. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *ICTL*, volume 827 of *Lecture Notes in Computer Science*, pages 480–505. Springer, 1994. (cited on p. 44)

[115] Nicoletta Fornara and Marco Colombetti. Operational specification of a commitment-based agent communication language. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pages 536–542. ACM, 2002. (cited on p. 49)

[116] J. France and A. A. Ghorbani. A multiagent system for optimizing urban traffic. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 411–414, 2003. (cited on pp. 69, 77, and 106)

[117] C. Frese, J. Beyerer, and P. Zimmer. Cooperation of cars and formation of cooperative groups. In *2007 IEEE Intelligent Vehicles Symposium*, pages 227–232, 2007. (cited on p. 82)

[118] Hans-Thomas Fritzsche. A model for traffic simulation. *Traffic Engineering and Control*, 35:317–321, 1994. (cited on p. 35)

[119] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 99, pages 548–553, 1999. (cited on p. 85)

[120] Nathan Gartner. Opac: A demand responsive strategy for traffic signal control. *Transportation Research Record Journal of the Transportation Research Board*, 1(906:75-81):75–81, January 1983. (cited on pp. 62 and 106)

[121] Nathan H. Gartner. Opac. *IFAC Control, Computers and Communication in Transportation Systems*, 1:241–244, 1989. (cited on pp. 62 and 106)

[122] L. Gasser, C. Bragamza, and N. Herman. MACE: a flexible testbed for distributed AI research. In M. Huhns, editor, *Distributed Artifical Intelligence*, pages 119–152. Pitman, London and Morgan Kaufmann, San Mateo, CA, 1987. (cited on p. 44)

[123] Les Gasser. Perspectives on organizations in multi-agent systems. In Michael Luck, Vladimir Marik, Olga Stepankova, and Robert Trappl, editors, *Multi-Agent Systems and Applications*, volume 2086 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2001. (cited on p. 50)

[124] Benjamin Gateau. *Modelisation et Supervision d'Institutions Multi-Agents.* PhD thesis, Ecole Nationale Superieure des Mines de Saint Etienne, June 2007. (cited on p. 50)

[125] Denos C. Gazis, Robert Herman, and Richard W. Rothery. Nonlinear follow-the-leader models of traffic flow. *Operations Research*, 9(4):545–567, 1961. (cited on p. 323)

[126] Michael R. Genesereth and Richard E. Fikes. Knowledge interchange format-version 3.0: Reference manual. Computer Science Department, Stanford University San Francisco, CA, 1992. (cited on p. 49)

[127] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In Kenneth D. Forbus and Howard E. Shrobe, editors, *AAAI*, pages 677–682. Morgan Kaufmann, 1987. (cited on p. 43)

[128] Carlos Gershenson. *Design and Control of Self-organizing Systems*. PhD thesis, Vrije Universiteit Brussel, 2007. (cited on pp. 69 and 106)

[129] Connie J. G. Gersick. Time and transition in work teams: Toward a new model of group development. *The Academy of Management Journal*, 31:9–41, 1988. (cited on p. 17)

[130] Joseph A. Giampapa and Katia P. Sycara. Team-oriented agent coordination in the retsina multi-agent system. *AGENTS02, July 15-19, 2002, Bologna, Italy.*, AGENTS02, 1:1–8, 2002. (cited on p. 92)

[131] David K. Gibson. Can we banish the phantom of traffic jam? intelligent cars will be able to head off congestion problems before they happen., 2016. (cited on p. 3)

[132] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15:105–111, 1981. (cited on pp. 28, 36, 105, 167, 247, and 248)

[133] P. G. Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403 – 414, 1986. (cited on pp. 31, 32, 36, 165, and 167)

[134] P. G. Gipps. Multism: A model for simulating vehicular traffic on multi-lane arterial roads. *Mathematics and Computers in Simulation*, 28:291–295, 1986. (cited on pp. 31 and 32)

[135] Davies Gleave, Roberta Frisoni, Andrea DallÓglio, Craig Nelson, James Long, Christoph Vollath, Davide Ranghetti, and Sarah McMinimy. Research for tran committee: Self-piloted cars: The future of road transport?, 2016. (cited on p. 2)

[136] Jorge J. Gómez-Sanz and Rubén Fuentes-Fernández. Understanding agent-oriented software engineering methodologies. *The Knowledge Engineering Review*, 30(4):375393, 2015. (cited on p. 9)

[137] Jana Görmer, Jan Fabian Ehmke, Maksims Fiosins, Daniel Schmidt, Henrik Schumacher, and Hugues Tchouankem. Decision support for dynamic city traffic management using vehicular communication. In Janusz Kacprzyk, Nuno Pina, and Joaquim Filipe, editors, *SIMULTECH*. SciTePress, 2011. (cited on pp. 110, 120, 122, 157, 199, 220, 232, 278, and 299)

[138] Jana Görmer, Gianina Homoceanu, Christopher Mumme, Michaela Huhn, and Jörg P. Müller. JREP: Extending repast simphony for jade agent behavior components. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, volume 2, pages 149–154, 2011. (cited on pp. 131, 220, 335, and 347)

[139] Jana Görmer and Jörg P. Müller. Group coordination for agent-oriented urban traffic management. In Yves Demazeau et al., editor, *Advances on Practical Applications of Agents and Multi-Agent Systems (Proc. of PAAMS 2012)*, pages 245–248. Springer, 2012. (cited on pp. 153, 157, and 220)

[140] Jana Görmer and Jörg P. Müller. Multiagent system architecture and method for group-oriented traffic coordination. In *Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on*, pages 1–6, 2012. (cited on pp. xvi, xvii, 137, 153, 157, 165, 168, 176, 178, 220, 242, and 244)

[141] Jana Görmer and Christopher Mumme. Multiagentensysteme für das kooperative Verkehrsmanagement. In Andrea Back, Markus Bick, Martin Breunig, Key Pousttchi, and Frédéric Thiesse, editors, *MMS 2012: Mobile und ubiquitäre Informationssysteme. Proceedings der 7. Tagung, 1. bis 2. März 2012 in Braunschweig, Deutschland*, pages 138–142, 2012. (cited on pp. 157, 176, and 220)

[142] Ilya Grigoryev. Anylogic 7 in three days: A quick course in simulation modeling. Amazon Digital Services, Inc, December 2014. (cited on p. 332)

[143] Patrick-Oliver Groß, Marlin W. Ulmer Jan F. Ehmke, and Dirk C. Mattfeld. Exploiting travel time information for reliable routing in city logistics. *Transportation Research Procedia*, 10:652 661, 2015. (cited on p. 251)

[144] Barbara Grosz and Sarit Kraus. *The Evolution of SharedPlans. Foundations of Rational Agencies,.* Kluwer Academic Press, 1999. (cited on p. 92)

[145] Barbara J. Grosz and Sarit Kraus. Collaborative plans for group activities. In *Proceedings of IJCAI-13*, volume Volume 1, pages pp. 367–373, Chambery, France, 1993. (cited on p. 92)

[146] Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. In *Artificial Intelligence*, number 86 in 2, pages pp. 269–357. Springer, 1996. (cited on pp. 90 and 92)

[147] Y. Guo and J. P. Müller. Multiagent collaborative learning for distributed business systems. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 1156–1163. ACM Press, July 2004. (cited on p. 137)

[148] Olivier Gutknecht and Jacques Ferber. Madkit: A generic multi-agent platform. In *Proceedings of the Fourth International Conference on Autonomous Agents*, AGENTS '00, pages 78–79. ACM Press, 2000. (cited on p. 50)

[149] Olivier Gutknecht and Jacques Ferber. Madkit. http://www.madkit.org/, 2017. (cited on p. 51)

[150] J. Richard Hackman. The design of work teams. *Handbook of organizational behavior*, 1:315–342, 1987. (cited on pp. xv, xvi, 16, 17, 18, 21, 159, 182, and 184)

[151] Randolph Hall and Chinan Chin. Vehicle sorting for platoon formation: Impacts on highway entry and throughput. *Transportation Research Part C: Emerging Technologies*, 13(56):405 – 420, 2005. (cited on p. 82)

[152] Simon Hallé. *Automated highway systems: Platoons of vehicles viewed as a multiagent system*. PhD thesis, 2005. (cited on p. 82)

[153] Mahdi Hannoun, Olivier Boissier, Jaime S. Sichman, and Claudette Sayettat. Moise: An organizational model for multi-agent systems. In Maria Carolina Monard and Jaime Simão Sichman, editors, *Advances in Artificial Intelligence*, volume 1952 of *Lecture Notes in Computer Science*, pages 156–165. Springer Berlin Heidelberg, 2000. (cited on p. 50)

[154] Hannes Hartenstein and Kenneth Laberteaux. *VANET - Vehicular Applications and Inter-Networking Technologies (Intelligent Transport Systems)*. John Wiley & Sons, 2010. (cited on pp. xvii, 21, 319, and 321)

[155] D. W. Harwood, K. M. Bauer, I. B. Potts, D. J. Torbic, K. R. Richard, E. R. Kohlman Rabbani, E. Hauer, and L. Elefteriadou. Safety effectiveness of intersection left- and right-turn lanes. Technical report, Midwest Research Institute Kansas City, 2002. (cited on p. 215)

[156] H. Haugeneder, D. Steiner, and F. G. McCabe. IMAGINE: A framework for building multi-agent systems. In S. M. Deen, editor, *Proceedings of the 1994 International Working Conference on Cooperating Knowledge Based Systems (CKBS-94)*, pages 31–64, 1994. (cited on p. 44)

[157] A. G. Hawkes. Gap-acceptance in road traffic. *Journal of Applied Probability*, 5:84–92, 1968. (cited on p. 32)

[158] Andrew J. Hawkins. Ubers self-driving cars are now picking up passengers in arizona, 2017. (cited on p. 2)

[159] Barbara Hayes-Roth, Lee Brownston, and Robert van Gent. Multiagent collaboration in directed improvisation. In *ICMAS*, pages 148–154, 1995. (cited on p. 37)

[160] Udo Heidl. Vissim 4.10 user manual, 2017. (cited on p. 132)

[161] Dirk Helbing. *Verkehrsdynamik*. Springer, 1997. (cited on p. 105)

[162] Dirk Helbing. Derivation of a fundamental diagram for urban traffic flow. *The European Physical Journal B*, 70(2):229–241, 2009. (cited on p. 168)

[163] Dirk Helbing, Stefan Lämmer, and Jean-Patrick Lebacque. Self-organized control of irregular or perturbed network traffic. In Christophe Deissenberg and Richard F. Hartl, editors, *Optimal Control and Dynamic Games*, volume 7 of *Advances in Computational Management Science*, pages 239–274. Springer US, 2005. (cited on p. 74)

[164] Horst Hellbrück, Axel Wegener, and Stefan Fischer. Autocast: A general-purpose data dissemination protocol and its application in vehicular networks. *Ad Hoc & Sensor Wireless Networks*, 6(1-2):91–122, 2008. (cited on p. 248)

[165] Bruce R. Hellinga. Requirements for the calibration of traffic simulation models. In *Proceedings of the Canadian Society for Civil Engineering Annual Conference*, number IVb, pages 211–222, 1998. (cited on pp. 276 and 277)

[166] J. J. Henry, J. L. Farges, and J. Tuffal. The prodyn real time traffic algorithm. In *IFAC Control in Transportation Systems*, pages 1–6, Baden-Baden, Federal Republic of Germany, 1983. (cited on pp. 62 and 106)

[167] Peter Hidas. A functional evaluation of the aimsun, paramics and vissim microsimulation models. *Road and Transport Research*, 14:45–59, 2005. (cited on p. 35)

[168] Koen V. Hindriks, Frank S. De Boer, Wiebe Van der Hoek, and John-Jules Ch. Meyer. Agent programming in 3apl. *Autonomous Agents and Multi-Agent Systems*, 2:357–401, 1999. Springer. (cited on p. 44)

[169] Petra Hirschmann. *Kooperative Gestaltung unternehmensübergreifender Geschäftsprozesse*. PhD thesis, Saarbrücken Universität, 1998. (cited on p. 48)

[170] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933. (cited on p. 272)

[171] Nick Howden, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas. JACK intelligent agents: Summary of an agent infrastructure. In *5th International Conference on Autonomous Agents*. ACM Press, 2001. (cited on p. 51)

[172] J. F. Hübner, J. S. Sichman, and O. Boissier. S-moise+: A middleware for developing organised multi-agent systems. In O. Boissier, V. Dignum, E. Matson, and J. S. Sichman, editors, *Proceedings of the International Workshop on Organizations in Multi-Agent Systems, from Organizations to Organization Oriented Programmin in MAS (OOOP'2005)*, volume 3913 of LNAI, pages 64–78. Springer, 2005. (cited on p. 50)

[173] Jomi F. Hübner, Olivier Boissier, Rosine Kitio, and Alessandro Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems*, 20(3):369–400, 2010. (cited on pp. 42, 52, 139, 141, and 230)

[174] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Advances in artificial intelligence*, pages 118–128. Springer, 2002. (cited on p. 50)

[175] Michaela Huhn, Aret Duraslan, Gianina Ganceanu, Jana Görmer, Jörg Hähner, Jörg P. Müller, Christian Müller-Schloer, Christopher Mumme,

and Christian Schulz. Autonomous agents in organized localities: meta-models and architecture. Technical Report NTH Technical Report 2010-02, Niedersächsischer Technische Hochschule, 2010. (cited on pp. 151, 157, and 159)

[176] Michaela Huhn, Jörg P. Müller, Jana Görmer, Gianina Homoceanu, N.-T. Le, Lukas Märtin, Christopher Mumme, Christian Schulz, Niels Pinkwart, and C. Müller-Schloer. Autonomous agents in organized localities regulated by institutions. In *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST)*, pages 54–61, 2011. (cited on pp. xvi, 151, 157, 159, 160, and 161)

[177] Michael N. Huhns and Larry M. Stephens. Multiagent systems and societies of agents. In Gerhard Weiss, editor, *Multiagent Systems*, chapter 2, pages 79–120. MIT Press, Cambridge, MA, USA, 1999. (cited on p. 99)

[178] Luke Hunsberger. *Group Decision Making and Temporal Reasoning.* PhD thesis, Harvard University, 2002. (cited on p. 85)

[179] P.B. Hunt, D.I. Robertson, R.D. Bretherton, and M.C. Rogle. The scoot on-line traffic signal optimization technique. In *Traffic Engineering and Control*, pages 190–199, 1982. (cited on pp. 59, 61, 77, 106, and 109)

[180] European Telecommunications Standards Institute. Intelligent transport systems (its); european profile standard for the physical and medium access control layer of intelligent transport systems operating in the 5 ghz frequency band. Electronic, 2010. (cited on pp. 58, 122, and 199)

[181] SAE INTERNATIONAL. Automated driving, 2014. (cited on p. 2)

[182] Sandy Irani and Vitus Leung. Scheduling with conflicts, and applications to traffic signal control. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '96, pages 85–94, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics. (cited on pp. 73 and 106)

[183] Sandy Irani and Vitus Leung. Probabilistic analysis for scheduling with conflicts. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, pages 286–295, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics. (cited on p. 106)

[184] ISO. Systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models. iso.org Website, March 2011. (cited on p. 111)

[185] Peter Jackson. *Introduction to Expert Systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1998. (cited on p. 38)

[186] Brian Slack Jean-Paul Rodrigue, Claude Comtois. *The Geography of Transport Systems*, volume 3. Routledge, 2013. (cited on p. 25)

[187] Nicholas R. Jennings. Towards a cooperation knowledge level for collaborative problem solving. In *ECAI*, pages 224–228, 1992. (cited on pp. 66, 80, and 106)

[188] Nicholas Robert Jennings. *Joint Intentions as a Model of Multi-Agent Cooperation in Complex Dynamic Environments.* PhD thesis, Department of Electronic Engineering, Queen Mary and Westfield College, University of London, 1992. (cited on p. 86)

[189] Nick R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8:223–250, 1993. (cited on p. 86)

[190] Nick R. Jennings. Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *Int. Journal of Intelligent and Cooperative Information Systems*, 2:289–318, 1993. (cited on p. 86)

[191] Nick R. Jennings. Coordination techniques for distributed artificial intelligence. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter Coordination Techniques for Distributed Artificial Intelligence, pages 187–210. John Wiley &amp; Sons, Inc., New York, NY, USA, 1996. (cited on pp. 45 and 170)

[192] Myounghoon Jeon, Bruce N. Walker, and Thomas M. Gable. The effects of social interactions with in-vehicle agents on a driver's anger level, driving performance, situation awareness, and perceived workload. *Applied Ergonomics*, 50:185 – 199, 2015. (cited on pp. 74 and 106)

[193] Bin Jiang, Xiao Xu, Chao Yang, Renfa Li, and Takao Terano. *Solving Road-Network Congestion Problems by a Multi-objective Optimization Algorithm with Brownian Agent Model*, pages 36–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. (cited on pp. 74 and 105)

[194] Andrew J. I. Jones and Xavier Parent. Conventional signalling acts and conversation. In *Advances in Agent Communication*, pages 1–17. Springer, 2004. (cited on p. 49)

[195] Andrew J. I. Jones and Xavier Parent. A convention-based approach to agent communication languages. *Group Decision and Negotiation*, 16:101–141, 2007. (cited on p. 49)

[196] Peter-J. Jost. *Organisation und Koordination - Eine konomische Einführung.* Gabler, 2 edition, 2009. (cited on p. 177)

[197] Robert Junges and Ana L. C. Bazzan. Evaluating the performance of dcop algorithms in a real world, dynamic problem. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, volume 2 of *AAMAS '08*, pages 599–606. International Foundation for Autonomous Agents and Multiagent Systems, 2008. (cited on pp. 77 and 105)

[198] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. (cited on p. 71)

[199] J. Kang, W. Kim, J. Lee, and K. Yi. Design, implementation, and test of skid steering-based autonomous driving controller for a robotic vehicle

with articulated suspension. *Journal of Mechanical Science and Technology*, 24(3):793–800, 2010. (cited on p. 65)

[200] R. Kates, K. Bogenberger, and M. Hoops. *Mesoscopic simulation with ANIMAL: Optimal utilization of downstream traffic detector data and the propagation of information.* Springer Germany, 1997. (cited on p. 21)

[201] J. O. Kephart and D. M. Chess. The vision of Autonomic Computing. *IEEE Computer*, 36:41–50, 2003. (cited on pp. 66 and 157)

[202] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobil for car-following models. *Transportation Research Record*, pages 86–94, 2007. (cited on pp. 31 and 33)

[203] A. Kieser and P. Walgenbach. *Organisation.* Schäffer-Poeschel, 5 edition, 2007. (cited on p. 177)

[204] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative, 1995. (cited on p. 37)

[205] F. Klügl, R. Herrler, and M. Fehler. Sesam: Implementation of agent-based simulation using visual programming. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 1439–1440, New York, NY, USA, 2006. ACM. (cited on p. 345)

[206] Franziska Klügl. *Multiagentensimulation: Konzepte, Werkzeuge, Anwendung.* Agententechnologie. Addison-Wesley, 2001. (cited on pp. 151, 199, 200, and 202)

[207] Franziska Klügl. Multiagent simulation - tutorial at easss2010. In *The 12th European Agent Systems Summer School*, pages 201–226, August 2010. (cited on pp. xv, 34, 35, 47, 50, 51, and 104)

[208] Kohlen. Dynamisches verkehrsmanagement in berlin. Powerpoint, 2014. (cited on pp. xv and 23)

[209] J. Kolodko and L. Vlacic. Cooperative autonomous driving at the intelligent control systems laboratory. *IEEE Intelligent Systems*, 18:8–11, July 2003. (cited on pp. 72 and 106)

[210] Iisakki Kosonen. Multi-agent fuzzy signal control based on real-time simulation. *Transportation Research Part C*, 11(5):389–403, oct 2003. (cited on p. 76)

[211] G. Kotushevski and K. A. Hawick. A review of traffic simulation software. Technical report, Computer Science, Massey University, 2009. (cited on p. 35)

[212] Jarosław Koźlak, Grzegorz Dobrowolski, Marek Kisiel-Dorohinicki, and Edward Nawarecki. Anti-crisis management of city traffic using agent-based approach. *Universal Computer Science*, 14(14):2359–2380, 2008. (cited on p. 76)

[213] Kraftfahrt-Bundesamt, May 2014. (cited on p. 270)

[214] D. Krajzewicz, M. Hartinger, G. Hertkorn, P. Mieth, C. Rössel, J. Zimmer, and P. Wagner. Using the road traffic simulation sumo for educational purposes. In Serge P. Hoogendoorn, Stefan Luding, Piet H. L. Bovy, Michael Schreckenberg, and Dietrich E. Wolf, editors, *Traffic and Granular Flow 2003*, pages 217–222. Springer Berlin Heidelberg, 2005. (cited on pp. 30 and 222)

[215] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012. (cited on pp. 3, 35, 36, and 139)

[216] Stefan Krauß. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. PhD thesis, Mathematisch Naturwissenschaftlichen Fakultät der Universität zu Köln, April 1998. (cited on pp. 21, 26, 28, 30, 31, 36, and 225)

[217] Charles Krome. 10 best self parking cars, 2017. (cited on p. 2)

[218] Jaeyoung Kwak. Development of calibration and validation procedure, and application of sustainable transportation system for transims traffic microsimulator. Technical report, CTS Virginia, 2009. (cited on p. 321)

[219] E. O. Postma L. J. P. van der Maaten and H. J. van den Herik. Dimensionality reduction: A comparative review. TiCC-TR 005, Tillburg University, 2009. (cited on p. 271)

[220] Yannis Labrou and Tim Finin. A semantics approach for kqml - a general purpose communication language for software agents. In *Proceedings of the third international conference on Information and knowledge management*, pages 447–455. ACM, 1994. (cited on p. 49)

[221] Yannis Labrou, Tim Finin, and Yun Peng. Agent communication languages: The current landscape. *IEEE Intelligent systems*, 14:45–52, 1999. (cited on p. 49)

[222] Andrew Lansdowne. *Traffic Simulation using Agent-Based Modelling*. BSc Computer Science. University of the West of England, 2006. (cited on p. 134)

[223] H. Laux and F. Liermann. *Grundlagen der Organisation: Die Steuerung von Entscheidungen als Grundproblem der Betriebswirtschaftslehre*. Springer-Lehrbuch. Springer, 6 edition, 2005. (cited on p. 177)

[224] California Legislative. Vehicles passing distance: Three feet for safety act. Electronic, September 2014. (cited on p. 323)

[225] Karsten Lemmer. Dlr institute of transportation systems. Website, 2001. (cited on p. 357)

[226] V. Lesser, K. Decker, N. Carver, A. Garvey, D. Neiman, M. Nagendra Prasad, and T. Wagner. Evolution of the gpgp domain-independent coordination framework. Technical report, University of Massachusetts Computer Science Technical Report 1998-05, 1998. (cited on p. 93)

[227] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. Nagendra Prasad, A. Raja, R. Vincent, P. Xuan, and X. Q. Zhang. Evolution of the gpgp/tms domain-independent coordination framework. In *Autonomous Agents and Multi Agent Systems*, Vol 9, page 87143. Kluwer Academic Publishers, 2004. (cited on p. 94)

[228] V. R. Lesser and L. D. Erman. Distributed interpretation: A model and and experiment. In *IEEE Transactions on Computers*, volume C-29, page 11441163, December 1980. (cited on p. 93)

[229] Hector J. Levesque, Philip R. Cohen, and José H. T. Nunes. On acting together. *AAAI*, 90:94–99, 1990. (cited on pp. 86 and 87)

[230] Andrey Lidokhover. Konzeption und Realisierung eines Teammodells für Multi-Agenten-Systeme. Master's thesis, Universität Hamburg Fachbereich Informatik, September 2005. (cited on pp. xv and 83)

[231] Jennie Lioris, Ramtin Pedarsani, Fatma Yildiz Tscikaraoglu, and Pravin Varaiya. Platoons of connected vehicles can double throughput in urban roads. Electronic TRB, November 27 2015. (cited on p. 251)

[232] Nicholas E. Lownes and Randy B. Machemehl. Vissim: A multi-parameter sensitivity analysis. In *Proceedings of the 38th Conference on Winter Simulation*, WSC '06, pages 1406–1413. Winter Simulation Conference, 2006. (cited on p. 35)

[233] P. R. Lowrie. *SCATS - Sydney Co-Ordinated Adaptive Traffic System - A Traffic Responsive Method of Controlling Urban Traffic.* Roads and Traffic Authority NSW, Darlinghurst, NSW Australia, Sydney, NSW, Australia, 1992. (cited on pp. 59, 61, 77, 106, and 109)

[234] M. Luck and M. d'Inverno. Engagement and cooperation in motivated agent modelling. In Zhang and Lukose, editors, *Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence, Lecture Notes in Artiffcial Intelligence*, volume 1087, pages 70–84. Springer-Verlag, 1996. (cited on pp. 80 and 131)

[235] Michael Luck and M. d'Inverno. A formal framework for agency and autonomy. In *ICMAS-95*, pages 254–60, 1995. (cited on p. 70)

[236] Marin Lujak, Holger Billhardt, and Sascha Ossowski. Distributed coordination of emergency medical service for angioplasty patients. *Annals of Mathematics and Artificial Intelligence*, 78(1):73–100, Sep 2016. (cited on p. 315)

[237] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, and Keith Sullivan. Mason: A new multi-agent simulation toolkit. In *In Proceedings of the 2004 Swarmfest Workshop (2004)*, 2004. (cited on p. 338)

[238] Grace Macaluso. A look back at 15 years of research and innovation at auto21, 2016. (cited on p. 2)

[239] T. W. Malone, K. G. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. S. Osborn, A. Bernstein, G. Herman, M. Klein, and E. ODonnell. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science*, 45:425–443, March 1999. (cited on p. 50)

[240] René Mandiau, Alexis Champion, Jean-Michel Auberlet, Stéphane Espié, and Christophe Kolski. Behaviour based on decision matrices for a coordination between agents in a urban traffic simulation. *Applied Intelligence*, 28 (2):121–138, 2008. (cited on pp. 72 and 106)

[241] C. E. Martin and K. S. Barber. Mutliple, simultaneous autonomy levels for agent-based systems. In *Proceedings Fourth Internation Conference on Control, Automation, Robotics, and Vision*, pages 1318–1322, Westing Stamford, Singapore, 1996. (cited on p. 70)

[242] Arne Kesting Martin Treiber. *Verkehrsdynamik und -simulation.* Springer, 2010. (cited on p. 35)

[243] Dirk Helbing Martin Treiber, Ansgar Hennecke. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62:1805–1824, 2000. (cited on p. 327)

[244] Enrique Martínez-García. Technological progress is key to improving world living standards, 2013. (cited on p. 3)

[245] Maja J. Matarić. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:51–80, 1995. (cited on pp. 80 and 105)

[246] V. Mauro and C. Di Taranto. Utopia. *IFAC Control, Computers and Communication in Transportation Systems*, 1(12):245–252, 1989. (cited on pp. 62 and 105)

[247] Joseph E. McGrath. *Social psychology: a brief introduction.* Holt, Rinehart and Winston, 1964. (cited on pp. xv and 21)

[248] R. M. Michaels. Perceptual factors in car following. In *International Symposium on the Theory of Road-Traffic Flow. Second. Proceedings*, OECD, pages 44–59, 1965. (cited on p. 324)

[249] John Miller. Self-driving car technologys benefits, potential risks, and solutions, 2014. (cited on p. 3)

[250] Owen Miller. Robotic cars and their new crime paradigms, 2014. (cited on p. 3)

[251] Pitu Mirchandani and Larry Head. A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies*, 9:415–432, 2001. (cited on pp. 63 and 106)

[252] Edgar Morin. *La Méthode, Tome 1: La Nature de la Nature*, volume 3. Seuil, 1977. (cited on p. 50)

[253] J. P. Müller. *The design of intelligent agents*, volume 1177 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1996. (cited on pp. 43, 157, 160, 162, 197, 198, 232, and 242)

[254] Jörg P. Müller. A cooperation model for autonomous agents. In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Intelligent Agents III Agent Theories, Architectures, and Languages*, volume 1193 of *Lecture Notes in Computer Science*, pages 245–260. Springer Berlin Heidelberg, 1997. (cited on p. 43)

[255] Jörg P. Müller and Klaus Fischer. Application impact of multi-agent systems and technologies: A survey. In Onn Shehory and Arnon Sturm, editors, *Agent-Oriented Software Engineering - Reflections on Architectures, Methodologies, Languages, and Frameworks*, pages 27–53. Springer, 2014. (cited on p. 8)

[256] Jörg P. Müller and Markus Pischel. The agent architecture interrap: concept and application. Technical report, Saarländische Universitäts- und Landesbibliothek, Postfach 151141, 66041 Saarbrücken, 1993. (cited on pp. 160, 164, and 167)

[257] Jörg P. Müller, Markus Pischel, and Michael Thiel. Modelling reactive behaviour in vertically layered agent architectures. In Michael Wooldridge and Nicholas R. Jennings, editors, *ECAI Workshop on Agent Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Computer Science*, pages 261–276. Springer, 1994. (cited on p. 43)

[258] C. Müller-Schloer, H. Schmeck, and T. Ungerer. *Organic Computing A Paradigm Shift for Complex Systems*. Autonomic Systems. Springer Basel, 2011. (cited on p. 7)

[259] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221–2229, 1992. (cited on pp. 42, 105, and 319)

[260] Srinivas Sankara Narayanan. *Karma: Knowledge-based Active Representations for Metaphor and Aspect*. PhD thesis, University of California, Berkeley, 1997. (cited on p. 50)

[261] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, USA, 1990. (cited on p. 89)

[262] Cynthia Nikolai and Gregory Madey. Tools of the trade: A survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, 12(2):2, 2009. (cited on p. 329)

[263] H.S. Nwana and D.T. Ndumu. An introduction to agent technology. In HyacinthS. Nwana and Nader Azarmi, editors, *Software Agents and Soft Computing Towards Enhancing Machine Intelligence*, volume 1198 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin Heidelberg, 1997. (cited on pp. xv, 41, and 46)

[264] Oliver Oberschelp, Thorsten Hestermeyer, Bernd Kleinjohann, and Lisa Kleinjohann. Design of self-optimizing agent-based controllers. In *Proceedings of the 3rd International Workshop on Agent-Based Simulation*, 2002. (cited on pp. 73 and 106)

[265] G.M.P. O'Hare and N. Jennings. *Foundations of Distributed Artificial Intelligence.* A Wiley-Interscience publication. Wiley, 1996. (cited on p. 66)

[266] J. Ortúzar and L. G. Willumsen. *Modelling Transport.* John Wiley & Sons, 3rd edition edition, 2001. (cited on pp. 68 and 105)

[267] Sascha Ossowski. *Agreement Technologies*, volume 8 of *Law, Governance and Technology Series.* Springer, 2013. (cited on p. vii)

[268] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:378–434, 2005. (cited on pp. 9 and 68)

[269] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1995. (cited on p. 298)

[270] Markos Papageorgiou, Christina Diakaki, Ioannis Nikolos, Ioannis Ntousakis, Ioannis Papamichail, and Claudio Roncoli. *Road Vehicle Automation 2*, chapter Freeway Traffic Management in Presence of Vehicle Automation and Communication Systems (VACS), pages 205–214. Springer International Publishing, Cham, 2015. (cited on pp. 161 and 210)

[271] Torsten O. Paulussen, Nicholas R. Jennings, Keith S. Decker, and Armin Heinzl. Distributed patient scheduling in hospitals. In *Proceedings 18th Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003. (cited on p. 94)

[272] G. Picard and M.-P. Gleizes. Cooperative self-organization to design robust and adaptive collectives. In *Second International Conference on Informatics in Control, Automation and Robotics (ICINCO'05), 14-17 September 2005, Barcelona, Spain, Volume I*, volume 1, pages 236–241. INSTICC Press, 2005. (cited on p. 183)

[273] Louis A. Pipes. An operational analysis of traffic dynamics. *Journal of Applied Physics*, 24(3):274–281, 1953. (cited on pp. 28 and 323)

[274] Tobias Pohlmann. *A New Method for Online Control of Urban Traffic Signal Systems.* PhD thesis, Technischen Universität Carolo-Wilhelmina zu Braunschweig, September 2010. (cited on pp. 64 and 315)

[275] Heribert Prantl. Autos von morgen, Recht von gestern, 2017. (cited on p. 1)

[276] Christian Priemer. *Konzept für eine kommunikationsdatenbasierte, dezentrale Lichtsignalsteuerung in städtischen Straßennetzen.* PhD thesis, Fakultät für Architektur, Bauingenieurwesen und Umweltwissenschaften der Technischen Universität Carolo-Wilhelmina zu Braunschweig, 2010. (cited on p. 64)

[277] Dr. Karl-Oskar Proskawetz. Cooperative road traffic: Forsight, safety and comfort. Electronic Internet, March 2016. (cited on p. 198)

[278] Holger Prothmann. *Organic Traffic Control.* PhD thesis, KIT, Fakulät für Wirtschaftswissenschaften, 2011. (cited on pp. 78 and 105)

[279] Holger Prothmann, Jürgen Branke, Hartmut Schmeck, Sven Tomforde, Fabian Rochner, Jörg Hähner, and Christian Müller-Schloer. Organic traffic light control for urban road networks. *International Journal of Autonomous and Adaptive Communications Systems*, 2(3):203–225, 2009. (cited on pp. 132 and 315)

[280] Holger Prothmann, Sven Tomforde, Jürgen Branke, Jörg Hähner, Christian Müller-Schloer, and Hartmut Schmeck. Organic traffic control. In Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing A Paradigm Shift for Complex Systems*, volume 1 of *Autonomic Systems*, pages 431–446. Springer Basel, 2011. (cited on p. 78)

[281] PTV. *VISSIM 5.40 Benutzerhandbuch*. epubli GmbH, 2012. (cited on pp. 3 and 35)

[282] Steven F. Railsback and Volker Grimm. *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton University Press, illustrated edition, 2011. (cited on p. 342)

[283] Steven F. Railsback, Steven L. Lytinen, and Stephen K. Jackson. Agent-based simulation platforms: Review and development recommendations. *Simulation*, 82(9):609–623, September 2006. (cited on p. 337)

[284] Prakash Ranjitkar, Takashi Nakatsuji, and Akira Kawamua. Car-following models: An experiment based benchmarking. *Journal of the Eastern Asia Society for Transportation Studies*, 6:1582 – 1596, 2005. (cited on pp. xvii, 28, and 320)

[285] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. et al. Allen, editor, *2nd Intern. Conf. on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1991. (cited on pp. 43 and 199)

[286] Anand S Rao. Agentspeak (l): Bdi agents speak out in a logical computable language. In Walter Van de Velde and John W. Perram, editors, *Agents Breaking Away*, volume 1038 of *LNCS*, pages 42–55. Springer, 1996. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), Eindhoven, The Netherlands, 22-25. (cited on p. 44)

[287] Anand S. Rao, Andrew Lucas, David Morley, Mario Selvestrel, and Graeme Murray. Agent-oriented architecture for aircombat simulation. Technical report, Australian Artificial Intelligence Institute, 1993. (cited on p. 37)

[288] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13(2):402–417, 1982. (cited on p. 85)

[289] Nedal T. Ratrout and Syed Masiur Rahman. A comparative analysis of currently used microscopic and macroscopic traffic simulation software. *Arabian Journal for Science & Engineering (Springer Science & Business Media BV)*, 34:121–131, 2009. (cited on p. 35)

[290] Wei Ren and Randal Beard. *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications.* Springer London, 2007. (cited on p. 82)

[291] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. "Give agents their artifacts": The A&A approach for engineering working environments in MAS. In Edmund Durfee, Makoto Yokoo, Michael Huhns, and Onn Shehory, editors, *6th International Joint Conference "Autonomous Agents & Multi-Agent Systems" (AAMAS 2007)*, pages 601–603, Honolulu, Hawai'i, USA, 14–18 May 2007. IFAAMAS. (cited on p. 144)

[292] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. CArtAgO: A framework for prototyping artifact-based environments in MAS. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Environments for MultiAgent Systems III*, volume 4389 of *LNAI*, pages 67–86. Springer, May 2007. 3rd International Workshop (E4MAS 2006), Hakodate, Japan, 8 May 2006. Selected Revised and Invited Papers. (cited on pp. 139 and 144)

[293] D.I. Robertson. Transyt: A traffic network study tool. Technical Report RL-253, Road Research Laboratory, Grothorne, Berkshire, England, 1969. (cited on pp. 59, 77, 106, and 109)

[294] Tom Robinson, Eric Chan, and Erik Coelingh. Operating platoons on public motorways: An introduction to the sartre platooning programme. In *17th world congress on intelligent transport systems*, volume 1, page 12, 2010. (cited on pp. 82, 300, 303, and 307)

[295] Stelios Rodoulis. The impact of autonomous vehicles on cities. *Journeys*, pages 12–20, 2014. (cited on pp. xv and 1)

[296] Danko A Roozemond. Using intelligent agents for urban traffic control systems. In *Proceedings of the international conference on artificial intelligence in transportation systems and science*, 1999. (cited on pp. 73 and 105)

[297] Danko A. Roozemond. Using intelligent agents for pro-active, real-time urban intersection control. *European Journal of Operational Research*, 131(2):293 – 301, 2001. ¡ce:title¿Artificial Intelligence on Transportation Systems and Science¡/ce:title¿. (cited on pp. 69, 73, and 105)

[298] R. J. F. Rossetti, P. A. F. Ferreira, R. A. M. Braga, and E. C. Oliveira. Towards an artificial traffic control system. In *Proceedings of the 11th IEEE Conference on Intelligent Transportation Systems (ITSC 2008)*, pages 14–19, Beijing, China, 2008. (cited on pp. 68 and 106)

[299] Rosaldo J. F. Rossetti, Rafael H. Bordini, Ana L. C. Bazzan, Sergio Bampi, Ronghui Liu, and Dirck Van Vliet. Using bdi agents to improve driver modelling in a commuter scenario. *Transportation Research Part C: Emerging Technologies*, 10:373 – 398, 2002. (cited on p. 142)

[300] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition).* SERIES IN ARTIFICIAL INTELLIGENCE. Pearson Education Limited, 3rd edition, 2014. (cited on pp. 8, 47, and 70)

[301] Mahnam Saeednia and Monica Menendez. A consensus-based algorithm for truck platooning. *IEEE*, 18:404–415, 2016. (cited on p. 3)

[302] N. Salazar, J. A. Rodriguez-Aguilar, J. L. Arcos, A. Peleteiro, and Juan C. Burguillo-Rial. Emerging cooperation on complex networks. In Sonenberg Tumer, Yolum and Stone, editors, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 669–676, 2011. (cited on pp. 80 and 131)

[303] David Sanderson, Didac Busquets, and Jeremy Pitt. A micro-meso-macro approach to intelligent transportation systems. In *IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 77–82. IEEE Computer Society, 2012. (cited on pp. xvi and 156)

[304] David William Sanderson. *Adaptation Strategies for Self-Organising Electronic Institutions*. PhD thesis, Imperial College London Department of Electrical and ElectronicnEngineering, May 2013. (cited on pp. xvi, 156, and 160)

[305] Tuomas W. Sandholm. *Distributed Rational Decision Making*, chapter 5, pages 201–258. MIT Press, 1999. (cited on p. 85)

[306] Roger Scharz, Anne Davidson, Peg Carlson, Sue McKinney, and Contributors. *The Skilled Facilitator Fieldbook: Tips, Tools, and Tested Methods for Consultants, Facilitators, Managers, Trainers, and Coaches*. Jossy-Bass - A Wiley Imprint, 1st edition, 2005. (cited on pp. 83, 177, and 178)

[307] Heiko Schepperle and Klemens Böhm. Agent-based traffic control using auctions. In Matthias Klusch, Koen V. Hindriks, Mike P. Papazoglou, and Leon Sterling, editors, *Cooperative Information Agents XI, 11th International Workshop, CIA 2007, Delft, The Netherlands, September 19-21, 2007, Proceedings*, volume 4676 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2007. (cited on pp. 81 and 105)

[308] Heiko Schepperle and Klemens Böhm. *Valuation-Aware Traffic Control: The Notion and the Issues*, chapter 10, pages 218–239. IGI Global, 2009. (cited on p. 81)

[309] H. Schmeck, C. Müller-Schloer, E. Cakar, M. Moez, and U. Richter. Adaptivity and self-organisation in organic computing systems. *ACM Transactions on Autonomous and Adaptive Systems*, 5(3):10:1–10:32, 2010. (cited on p. 65)

[310] Thomas Schulze and Thomas Fliess. Urban traffic simulation with psychophysical vehicle-following models. In *Proceedings of the 29th conference on Winter simulation*, pages 1222–1229. IEEE Computer Society, 1997. (cited on p. 324)

[311] Christiane Schulzki-Haddouti. Bundestag verabschiedet gesetz zum autonomen fahren, 2017. (cited on p. 1)

[312] Thomas Schwerdtfeger. *Makroskopisches Simulationsmodell für Schnellstraßennetze mit Berücksichtigung von Einzelfahrzeugen (DYNEMO)*. PhD thesis, University of Karsruhe, Germany, 1987. (cited on p. 21)

[313] Task Group SCI-144. Integration of systems with varying levels of autonomy. Technical report, North Atlantic Treaty Organisation, Research and Technology Organisation, 2008. (cited on p. 153)

[314] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969. (cited on p. 49)

[315] John R. Searle. *Collective intentions and actions*, volume 401. MIT Press Cambridge (Mass.), 1990. (cited on pp. 86 and 88)

[316] E. Semsar-Kazerooni and K. Khorasani. *Team Cooperation in a Network of Multi-Vehicle Unmanned Systems: Synthesis of Consensus Algorithms*. SpringerLink : Bücher. Springer New York, 2012. (cited on p. 82)

[317] Guni Sharon and Peter Stone. A protocol for mixed autonomous and human-operated vehicles at intersections. In *Proceedings of the 2nd International Workshop on Agent-based modeling of urban systems (ABMUS 2017)*, 2017. (cited on p. 72)

[318] Yoav Shoham. Agent-oriented programming. *Artif. Intell.*, 60(1):51–92, March 1993. (cited on p. 44)

[319] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 1st edition, 2009. (cited on p. 198)

[320] Munindar P. Singh. Social and psychological commitments in multiagent systems. In *In AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, pages 104–106. AAAI Inc., 1991. (cited on p. 49)

[321] Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE computer*, 31:40–47, 1998. (cited on p. 48)

[322] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113, December 1980. (cited on p. 99)

[323] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In *Distributed Artificial Intelligence*, pages 357–366. Kaufmann Publishers Inc., San Francisco, CA. USA, 1988. (cited on p. 99)

[324] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems Man and Cybernetics*, 11(02):61–70, January 1981. (cited on p. 99)

[325] John A. Sokolowski and Catherine M. Banks. *Modeling and Simulation Fundamentals: Theoretical Underpinningsand Practical Domains*. Wiley, 2010. (cited on p. 35)

[326] Matthijs Spaan. Team play among soccer robots. Master's thesis, Artificial Intelligence University of Amsterdam, April 2002. (cited on pp. 87 and 93)

[327] Merlijn Steingröver, Roelant Schouten, Stefan Peelen, Emil Nijhuis, and Bram Bakker. Reinforcement learning of traffic light controllers adapting

to traffic congestion. In *Proceedings of the Belgium-Netherlands Artificial Intelligence Conference, BNAIC05*, 2005. (cited on pp. 79 and 105)

[328] John Sterman. *Business Dynamics: Systems Thinking and Modeling for a Complex World.* McGraw Hill, 2000. (cited on p. 332)

[329] Christoph Stiller, Georg Färber, and Sören Kammel. Cooperative cognitive automobiles. In *Proceedings of the 2007 IEEE Intelligent Vehicle Symposium, Istanbul, Turkey June 13-15, 2007*, pages 215–220. IEEE, June 2007. (cited on pp. xv, 33, 34, and 157)

[330] Burkhard Straßmann. Designer, bitte an die arbeit, 2017. (cited on p. 1)

[331] Cédric Sueur, Jean-Louis Deneubourg, and Odile Petit. From social network (centralized vs. decentralized) to collective decision-making (unshared vs. shared consensus). *PLoS ONE*, 7:e32566, 2012. (cited on pp. 80 and 106)

[332] Andrew J. Sullivan, Naveen Cheekoti, Michael D. Anderson, and Dillip Malave. *Traffic simulation software comparison study*, volume 2217. UTCA, 2004. (cited on p. 134)

[333] Eric Sundstrom, Kenneth E De Meuse, and David Futrell. Work teams: Application and effectiveness. *American Psychologist*, 45:120–133, February 1990. (cited on pp. xvi and 159)

[334] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, USA, 1998. (cited on p. 71)

[335] Milind Tambe. Towards flexible teamwork. *J. Artif. Intell. Res. (JAIR)*, 7:83–124, 1997. (cited on pp. 80, 84, 91, 93, and 97)

[336] Milind Tambe, W. Lewis Johnson, Olph M. Jones, Frank Koss, John E. Laird, Paul S. Rosenbloom, and Karl Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16:15–39, 1995. (cited on p. 37)

[337] Joseph Tardo and Luis Valente. Mobile agent security and telescript. In *IEEE Computer Society*, pages 58–63, 1996. (cited on p. 44)

[338] H. Tchouankem, D. Schmidt, and H. Schumacher. Impact of vehicular communication performance on travel time estimation in urban areas. In *6th International Symposium "Networks for Mobility 2012"*, Stuttgart, Germany, September 2012. (cited on pp. 122 and 232)

[339] Hugues Narcisse Tchouankem. Radio shadowing in vehicle-to-vehicle communication at urban intersections - a measurement and simulation-based evaluation, 2016. (cited on p. 232)

[340] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White. A multi-agent systems approach to autonomic computing. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1 of *AAMAS'04*, pages 464–471, Washington, DC, USA, 2004. IEEE Computer Society. (cited on p. 66)

[341] Stephanie Teufel, Christian Sauter, and Kurt Bauknecht. *Computerunterstützung für die Gruppenarbeit.* Addison-Wesley, 1997. (cited on p. 48)

[342] S. Rebecca Thomas. The placa agent programming language. In Michael J. Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents*, volume 890 of *Lecture Notes in Computer Science*, pages 355–370. Springer Berlin Heidelberg, 1995. (cited on p. 44)

[343] X. Q. Tian and Y. D. Xu. Traffic network equilibrium problems with capacity constraints of arcs and linear scalarization methods. *Journal of Applied Mathematics*, 2012:12, 2012. (cited on p. 173)

[344] Ernest Peter Todosiev. The action point model of the driver-vehicle system. Technical report, Ohio State University. Antenna Laboratory, 1963. 266 pages. (cited on p. 324)

[345] T. Toledo, H. N Koutsopoulos, and M. E Ben-Akiva. Modeling integrated lane-changing behavior. *Transportation Research Record*, 1857(-1):30–38, 2003. (cited on p. 32)

[346] Sven Tomforde. *An Architectural Framework for Self-configuration and Self-improvement at Runtime.* PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, September 2011. (cited on pp. 64 and 78)

[347] W. S. Torgerson. Multidimensional scaling i: Theory and method. *Psychometrika*, 17:401–419, 1952. (cited on p. 272)

[348] Martin Treiber, Arne Kesting, and Christian Thiemann. *Traffic Flow Dynamics: Data, Models and Simulation.* Springer, 2013. (cited on pp. 24, 35, and 120)

[349] Bruce W. Tuckman. Developmental sequence in small groups. *Psychological Bulletin*, 63:384–399, 1965. (cited on pp. 19, 191, and 192)

[350] Bruce W. Tuckman and Mary Ann C. Jensen. Stages of small-group development revisited. *Group & Organization Management*, 2:419–427, 1977. (cited on pp. xv, 19, 190, 191, and 193)

[351] K. Tumer, A. K. Agogino, and Z. Welch. Traffic congestion management as a learning agent coordination problem. In A. Bazzan and F. Klügl, editors, *Multiagent Architectures for Traffic and Transportation Engineering*, pages 261–279. Lecture notes in AI, Springer, 2009. (cited on pp. 82 and 105)

[352] Kagan Tumer, Zachary T. Welch, and Adrian K. Agogino. Aligning social welfare and agent preferences to alleviate traffic congestion. In Lin Padgham, David C. Parkes, Jrg P. Mller, and Simon Parsons, editors, *AAMAS 2*, pages 655–662. IFAAMAS, 2008. (cited on p. 81)

[353] Douglas P. Twitchell, Mark Adkins, Jay F. Nunamaker Jr., and Judee K. Burgoon. Using speech act theory to model conversations for automated classification and retrieval. In *Proceedings of the 9th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2004)*, pages 121–130, 2004. (cited on p. 49)

[354] Berkeley. University of California. Path research report. Technical report, Institute of Transportation Studies., 1988. (cited on pp. 2 and 3)

[355] Kerrie L. Unsworth and Michael A. West. Teams: the challenges of co-operative work. In Nik Chmiel, editor, *Introduction to Work and Organizational Psychology - a European Perspective*, chapter 14, pages 327–346. Blackwell Publishing Ltd, 2000. (cited on p. 18)

[356] Chris Urmson. Google makes the case for a hands-off approach to self-driving cars, 2016. (cited on p. 2)

[357] Reich v Long. Court case california. Internet, May 1950. (cited on p. 323)

[358] L. J. P. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, November 2008. (cited on pp. 271 and 272)

[359] R. van Katwijk and P. van Koningsbruggen. Coordination of traffic management instruments using agent technology. *Transport Research Part C: Engineering Technologies*, 10:455–471, 2002. (cited on pp. 69 and 106)

[360] R. van Katwijk, P. van Koningsbruggen, B. De Schutter, and J. Hellendoorn. *Software Agent Technologies and Autonomic Computing*, chapter Applications of Agent Technology in Traffic and Transportation, pages 113–131. Whitestein Series. Birkhäuser, 2005. (cited on pp. 69 and 105)

[361] Pravin Varaiya. Smart cars on smart roads: problems of control. *IEEE*, pages 195–207, 1993. (cited on p. 82)

[362] Matteo Vasirani. *Vehicle-centric coordination for urban road traffic management - a market-based multi-agent approach.* PhD thesis, Universidad Rey Juan Carlos, 2009. (cited on pp. 68, 110, and 198)

[363] Matteo Vasirani and Sascha Ossowski. A market-inspired approach to reservation-based urban road traffic management. In Carles Sierra, Cristiano Castelfranchi, Keith S. Decker, and Jaime Simão Sichman, editors, *AAMAS (1)*, volume 1, pages 617–624. IFAAMAS, 2009. (cited on pp. 68, 106, and 151)

[364] Matteo Vasirani and Sascha Ossowski. A computational market for distributed control of urban road traffic systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):313–321, 2011. (cited on p. 68)

[365] Daniel Villatoro and Jordi Sabater-Mir. Towards the group formation through social norms. In *Sixth European Workshop on Multi-Agent Systems (EUMAS08)*, pages 1–15, 2008. (cited on p. 84)

[366] Tom Wagner, Valerie Guralnik, and John Phelps. A key-based coordination algorithm for dynamic readiness and repair service coordination. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 03), ACM Press*, pages 757–764. ACM, 2003. (cited on p. 94)

[367] Tom Wagner, Valerie Guralnik, and John Phelps. Taems agents: Enabling dynamic distributed supply chain management. *Electronic Commerce Research and Applications*, 2(2):114–132, 2003. (cited on pp. 94 and 95)

[368] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992. (cited on p. 310)

[369] C. E Wallace, K. G. Courage, D. P. Reaves, G. W. Schoene, G. W. Euler, and A. Wilbur. *Transyt-7F User's Manual*. University of Florida, 2003. (cited on p. 59)

[370] F.-Y. Wang. Toward a revolution in transportation operations: Ai for complex systems. In *IEEE Intelligent Systems*, volume 6, pages 8–13, 2008. (cited on pp. 67 and 105)

[371] Axel Wegener, Horst Hellbrück, Stefan Fischer, Björn Hendriks, Christiane Schmidt, and Sándor P. Fekete. Designing a decentralized traffic information system - autonomos. In *Kommunikation in Verteilten Systemen (KiVS), 16. Fachtagung Kommunikation in Verteilten Systemen (KiVS 2009), Kassel, 2.-6. März 2009, Eine Veranstaltung der Gesellschaft für Informatik (GI) unter Beteiligung der Informationstechnischen Gesellschaft (ITG/VDE) Ausgerichtet von der Universität Kassel*, pages 309–315, 2009. (cited on p. 248)

[372] Axel Wegener, Elad M. Schiller, Horst Hellbrück, Sándor P. Fekete, and Stefan Fischer. Hovering data clouds: A decentralized and self-organizing information system. In Hermann de Meer and James P. G. Sterbenz, editors, *Self-Organizing Systems, First International Workshop, IWSOS 2006, and Third International Workshop on New Trends in Network Architectures and Services, EuroNGI 2006, Passau, Germany, September 18-20, 2006, Proceedings*, volume 4124 of *Lecture Notes in Computer Science*, page 243247, 2006. (cited on p. 248)

[373] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, 1999. (cited on pp. xv, xv, 37, 40, 42, 45, 46, 47, 49, and 170)

[374] Gerhard Weiss. *Multiagent Systems*. MIT Press, 2013. (cited on p. 42)

[375] Michael A. West. *Handbook of Work Group Psychology*. Wiley, 1996. (cited on p. 14)

[376] Michael A. West. *International Handbook of Organizational Teamwork and Cooperative Working*. Wiley, 2003. (cited on pp. 16 and 17)

[377] R. Wiedemann. Simulation des Straßenverkehrsflusses. Technical report, Schriftenreihe des Instituts für Verkehrswesen der Universität Karlsruhe, 1974. (cited on p. 324)

[378] Marco Wiering. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158, 2000. (cited on pp. 79 and 105)

[379] M. J. Wooldridge and N. R. Jennings. Towards a theory of cooperative problem solving. In *6th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-94)*, pages 15–26, 1994. (cited on pp. 80 and 131)

[380] Michael Wooldridge and Nicholas R. Jennings. Agent theories, architectures, and languages: A survey. In Michael J. Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents*, volume 890 of *Lecture Notes in Computer Science*, pages 1–39. Springer Berlin Heidelberg, 1995. (cited on p. 44)

[381] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995. (cited on p. 40)

[382] Michael J. Wooldridge. *An Introduction to MultiAgent Systems.* John Wiley and Sons, 2nd edition, 2009. (cited on pp. xv, 36, 40, 42, 46, 68, 86, 98, 112, and 153)

[383] Rolf P. Würtz. *Organic Computing.* Understanding Complex Systems. Springer, 2008. (cited on pp. 66 and 78)

[384] Hiroaki Yamaguchi, Tamio Arai, and Gerardo Beni. A distributed control scheme for multiple robotic vehicles to make group formations. *Robotics and Autonomous Systems*, 36(4):125 – 147, 2001. (cited on pp. 184 and 223)

[385] T. Yamashita and K. Kurumatani. New approach to smooth traffic flow with route information sharing. In Ana L. C. Bazzan and Franziska Klügl, editors, *Multi-Agent Systems for Traffic and Transportation*, pages 291–306. IGI Global, 2009. (cited on pp. 82 and 105)

[386] Xuefeng Yan, Feng Gu, Xiaolin Hu, and Carl Engstrom. Dynamic data driven event reconstruction for traffic simulation using sequential monte carlo methods. In *Simulation Conference (WSC), 2013 Winter*, pages 2042–2053. IEEE, 2013. (cited on p. 35)

[387] Q. Yang and H. N. Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3):113–129, 1996. (cited on p. 32)

[388] Maicon R. Zatelli and Jomi Fred Hübner. A unified interaction model with agent, organization, and environment. In Emerson Cabrera Paraiso, Julio Cesar Nievola, and Renato Tinós, editors, *Anais do 9 Encontro Nacional de Inteligncia Artificial (ENIA 2012)*, 2012. (cited on pp. xv and 39)

[389] Bernard P. Zeigler. *Theory of Modelling and Simulation.* Wiley, 1976. (cited on p. 152)

[390] Benjamin Zhang. Autonomous cars could save the us 1.3 $trillion\ dollars\ a\ year$, 2014.(*cited on p.* 3)

Curriculum Vitae

Jana Görmer-Redding was born in Berlin, Germany, on November 4, 1980. At the age of fifteen, she moved to the region of the southern Harz in the district of Göttingen, in Lower Saxony, Germany, where she lived until completing secondary education. From 2000 to 2005 she studied at the Technical University of Clausthal (TUC), Germany, majoring in Information Systems. Before her graduation Jana joined Homanit GmbH and Co KG (i.a. products for automotive industry), Herzberg, Germany, for a working student in quality management. Furthermore, she worked as a student assistant at the Technical University of Clausthal.

Jana freelanced as a consultant for quality and environmental management for KKT Holding GmbH (i.a. products for automotive industry), Osterode, Germany. For her graduation project on "SCOTI: stable and reliable communication over the Internet for geographically distributed clusters" Jana spent a semester at Autonomous University of Barcelona (UAB), Bellaterra, Barcelona, Spain joining a research collaboration in 2003-2004.

From 2006 to 2009 Jana was employed as a researcher and project manager at the Clausthaler Umwelttechnik-Institut GmbH (CUTEC), a non-university research institution of Lower Saxony. From 2009 to 2015 Jana was employed as a researcher and lecturer at the Technical University of Clausthal, working on her dissertation in the information systems group. In Mai/June 2012 Jana was investigating in a short term scientific mission at Centre for Intelligent Information Technologies, University Rey Juan Carlos Mostoles, Spain. In 2016 Jana was employed as senior consultant and IT project manager for energy systems at EWERK, Leipzig, Germany. 2017 she was working as team leader in accounting and finance and managing ERP implementation projects at GISA GmbH, Halle, Germany. Today (2018) she is taking certified project management courses in Prince2, ITIL and Scrum and will use that knowledge as a program manager with PDV Systeme GmbH, Erfurt. Jana completed her doctoral studies on "Autonomous Vehicle Groups in Urban Traffic" early 2018.