An Architectural Framework for Self-configuration and Self-improvement at Runtime

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

von M.Sc. Sven Tomforde

geboren am 17. Dezember 1978 in Buchholz i.d. Nordheide

2011

1. Referent: Prof. Dr.-Ing. Christian Müller-Schloer
2. Referent: Prof. Dr. Pedro José Marrón
Tag der Promotion: 23.09.2011

"42" – Douglas Adams

Zusammenfassung

Schlagworte: Organic Computing, Framework, Selbstverbesserung, Selbstkonfiguration, sicherheitsorientiertes Lernen, Machinelles Lernen, Verkehrssteuerung, Netzwerkprotokoll, Learning Classifier Systeme

Stellen Sie sich eine Welt vor, in der Fahrzeuge ohne einen menschlichen Fahrer zu benötigen autonom fahren können, in der Kühlschränke für die Versorgung mit Lebensmitteln des täglichen Bedarfs verantwortlich sind, in der sich Produktionspläne automatisch an sich ändernde Bedürfnisse anpassen können bevor überhaupt eine Änderung des Bedarf beobachtet wird oder in der ein Team von Fußballrobotern die menschliche Weltmeistermannschaft besiegen kann. Dies wäre eine Welt, in der auf der einen Seite technische Geräte wirklich den Nutzern dienen, indem sie sich kontinuierlich an sich ändernde Gegebenheiten anpassen. Auf der anderen Seite wären diese Geräte auf eine bestimmte Art auch ein emanzipierter Teil unserer Gesellschaft.

Einige dieser aufgeführten Ideen sind inzwischen näher an der Marktreife angelangt als man sich vielleicht vorstellt. Beispielsweise betreibt *Google* bereits jetzt autonome Fahrzeuge im regulären Straßenverkehr, Kühlschränke können bereits von außerhalb überwacht werden und Produktionspläne passen sich zumindest an die aktuellen Gegebenheiten an. Basis all dieser technischen Systeme – sowohl der existierenden als auch der aufkommenden neuen – ist einerseits eine geeignete Sensortechnologie, um die aktuelle Situation wahrnehmen zu können, und andererseits eine Adaptionslogik, die in der Lage ist, auf die wahrgenommenen Stimuli angemessen zu reagieren.

Dieser Aspekt der *Adaption* ist das Hauptthema dieser Arbeit. Klassischerweise lässt sich Forschung in Industrie und Wissenschaft entlang von zwei Richtungen einordnen: Entweder wird versucht, das Potential und die Fähigkeiten existierender Systeme weiterzuentwickeln, oder es wird versucht, neue Visionen zu etablieren und Prototypen für diese aufzubauen. Eine mindestens ebenso herausfordernde Richtung ist es, beide Optionen zu kombinieren. Warum sollten wir darauf warten, dass neue adaptive Lösungen marktreif sind und sich im täglichen Bedarf durchsetzen? Stattdessen ist es doch wesentlich vielversprechender, aufkommende Visionen und neue Paradigmen in der Entwicklung von technischen Systemen mit existierendem Wissen und etablierten Lösungen zu kombinieren. Ein Weg, dieses umzusetzen, wird im Rahmen dieser Arbeit beschrieben.

Daher präsentiert diese Arbeit einen grundlegenden Entwurf für Systeme sowie das darauf aufbauende Framework, anhand dessen Eigenschaften wie Selbstkonfiguration und Selbstverbesserung für parametrisierbare technische Systeme zur Laufzeit ermöglicht werden. Ein erwartetes Ergebnis durch die Anwendung des Frameworks auf existierende Systeme ist die Herbeiführung gewünschter Eigenschaften wie Adaptivität und Robustheit. Aufbauend auf dem generellen Systementwurf werden im Rahmen dieser Arbeit Möglichkeiten zur Anwendung maschineller Lerntechniken in Echtweltsystemen untersucht. Dazu wer-

den zwei neue Varianten von Learning Classifier Systemen und Fuzzy Classifier Systemen entwickelt. Diese beiden modifizierten Ansätze werden in das Framework integriert und stellen dabei einen wichtigen Mechanismus zur Realisierung der Selbstverbesserungseigenschaften dar.

Die wissenschaftlichen Erkenntnisse dieser Arbeit werden im Folgenden benannt. Eingangs wird das bereits angesprochene Framework entwickelt und vorgestellt. Dieses Framework ist in der Lage, den Selbstkonfigurationsprozess im laufenden Betrieb autonom zu verbessern, indem Techniken des maschinellen Lernens eingesetzt werden. Daraufhin wird das spezifische Lernproblem, das von einem Teil des Frameworks definiert wird, klassifiziert. Aufbauend darauf wird nach passenden Techniken zu Lösung dieser Problematik gesucht. Dazu werden dann die bereits benannten Varianten existierender maschineller Lernverfahren entwickelt und vorgestellt. Weiterhin stellt diese Arbeit wesentliche neue Verfahren für zwei weitere Forschungsgebiete vor, wobei beide Ansätze auf dem entwickelten Framework beruhen. Im Bereich der Straßenverkehrsforschung wird ein System eingeführt, das die Freigabezeiten an innerstädtischen Ampelanlagen automatisch an sich ändernde Verkehrssituationen anpasst. Das gleiche Framework dient dann im Bereich der Datenkommunikation dazu, ein System zu entwickeln, das die Protokollkonfigurationen von Datenkommunikationsprotokollen dynamisch und zur Laufzeit an beobachtete Situationen in der Umgebung des Knotens anpassen kann. Abschließend wird ein abstrahiertes Modell zur Klassifizierung von Echtweltsystemen eingeführt, anhand dessen die Menge an Systemen identifiziert werden kann, für die das Framework einsetzbar ist.

Untersuchungen haben gezeigt, dass technische Systeme, die über das zusätzliche Framework verfügen, eine erheblich bessere Systemleistung erzielen können, wie herkömmliche Systeme. Als Beispiel dienen die beiden exemplarisch untersuchten Hauptanwendungen dieser Arbeit: Einerseits kann die verfügbare Kommunikationskapazität in mobilen ad-hoc Netzwerken um etwa 6 % erhöht werden, während andererseits die auftretenden Verlustzeiten von Fahrzeugen in städtischen Verkehrsnetzen um signifikante 16 % reduziert werden können – jeweils in Abhängigkeit von den untersuchten Szenarien. Weiterhin zeigen die Ergebnisse der ebenfalls durchgeführten generalisierten Untersuchung, dass das Framework in der Lage ist, seiner Zielsetzung auch unter Einflüssen wie Störungen und verrauschten Sensordaten nachzukommen.

Abstract

Keywords: Organic Computing, Framework, self-improvement, self-configuration, safety-oriented learning, machine learning, traffic control, network protocols, learning classifier systems

Imagine a world where cars drive autonomously without the need for a driver, fridges are responsible for keeping the needed amount of food, work schedules re-organise themselves automatically considering the correct priorities, production systems change their production schedule before the observed demands change or even shortages occur, or a soccer team consisting of robots beats the (human) world champion team. This would be a world where, on the one hand, technical devices really *serve* their users by adapting continuously to changing conditions and demands, and, on the other hand, become a somehow emancipated part of the world.

Some of these ideas are closer to market maturity as one might assume. For instance, *Google* already operates autonomous cars, fridges can be supervised remotely already, smart homes are on the way to become a part of reality, and production systems adapt at least to current conditions. Basis of all these technical systems – both, existing and upcoming ones – is (a) an appropriate sensor technology capable of detecting the situation and (b) an adaptation logic capable of appropriately reacting on these stimuli.

This *adaptation* aspect is the major topic of this thesis. Typically, research by industry and science can be categorised among two main research directions: increase the potential and the abilities of existing systems or establish new visions and develop prototypes. But an equally challenging direction is to make way for a cooperation of both approaches. Why should we wait until new adaptive solutions will be well-engineered and well-established? A better concept is to somehow combine upcoming visions and new paradigms of system design with existing technical knowledge and solutions. This *somehow* is the common theme of this thesis.

Therefore, this thesis presents the system design and the corresponding framework to enable capabilities like self-configuration and self-improvement for parametrisable systems at runtime. As a result of these capabilities, systems equipped with the framework as additional control mechanism are characterised by aspects like adaptivity and robustness. Besides the general system design, the thesis investigates the possibility of applying machine learning techniques to real-world applications – two novel variants of Learning Classifier Systems and Fuzzy Classifier Systems are developed. These modified machine learning techniques are integrated into the framework. Thereby, they take over the responsibility of the self-improvement tasks.

The contribution to scientific knowledge presented in this thesis is given as follows. Initially, the architectural framework is developed. This framework is capable of self-improving the reconfiguration behaviour autonomously by making use of machine learning techniques.

In addition, this thesis characterises the specific learning problem and investigates which techniques are applicable. Additionally, modified variants for the most promising techniques are developed to cover the restrictions and requirements of real-world systems and their safety-demands. Furthermore, the thesis presents two contributions to the state of the art in traffic control systems and data communication – both based on the general design. In traffic control, a novel decentralised system to adapt traffic control strategies at urban intersections according to changes in the traffic conditions is presented. The same system design applied to data communication results in a locally-organised system to reconfigure network protocol parameter sets as response to observed situations. Finally, the thesis introduces a generalised model to classify real-world systems that are controllable by the framework.

Analytical considerations of the evaluation results demonstrate the benefit of applying the developed framework to the control of technical systems. For instance, the available communication bandwidth in mobile ad-hoc networks can be increased by about 6 %, while the delays of vehicles in urban road networks can be dramatically decreased by up to 16 % depending on the investigated scenario. In addition, the results of the generalised investigation show that the framework is able to fulfil its task even under challenging conditions such as noise and disturbances.

List of Abbreviations

3LA	Three-layered Architecture
AAA	Adaptive Agent Architecture
AC	Autonomic Computing
ACT	Agreed Cycle Time
ADRA	Adaptive Distributed Resource Allocation Scheme
AE	Autonomous Element
AID	Automated Incident Detection
ALARM	Adaptive Location Aided Routing from Mines
AM	Autonomic Manager
ANN	Artificial Neural Network
AOC	Autonomy Oriented Computing
AP	Averaged Performance
AS	Acceptance Space
BDI	Belief-Desire-Intention
CA	California Algorithm
CM	Control Mechanism
CS	Configuration Space
COUGAAR	Cognitive Agent Architecture
DARPA	Defense Advanced Research Projects Agency
DCT	Desired Cycle Time
DE	Differential Evolution
DNF	Disjunctive Normal Form
DPSS	Decentralised Progressive Signal System
DS	Dead Space
DVR	Distance Vector Routing
EA	Evolutionary Algorithm
FCD	Floating Car Data
FCS	Fuzzy Classifier System
FIFA	Federation Internationale de Football Association
FSM	Finite State Machine
FTC	Fixed-time Controller
GA	Genetic Algorithm
GPS	Global Positioning System
HM	Harmony Memory
HMCR	Harmony Memory Consideration Rate
HPSS	Hierarchical Progressive Signal System
HS	Harmony Search
IP	Internet Protocol

IT	Information Technology
ITS	Intelligent Transportation System
J2EE	JAVA 2 Enterprise Edition
LCS	Learning Classifier System
LOS	Level of Service
LSR	Link State Routing
MAC	Media Access Control
MANet	Mobile Ad-hoc Network
MAPE	Monitor-Analyse-Plan-Execute
MARVIN	MANet Relative Velocity Indicator
MAS	Multi-Agent System
MASON	Multi-Agent Simulator of Neighbourhoods
ML	Machine Learning
MPC	Model Predictive Control
MR	Managed Resource
NACK	Not Achknowledged
NC	Number of Evaluation Function Calls
NEMA	National Electrical Manufacturers Association
NTP	Network Time Protocol
OAA	Open Agent Architecture
OC	Organic Computing
OCOM	Organic Computing in Off-highway Machines
ONC	Organic Network Control
OSOA	Organic Service-oriented Architecture
OTC	Organic Traffic Control
P2P	Peer-to-Peer
PC	Personal Computer
PID	Proportional-Integral-Derivative
PSO	Particle Swarm Optimisation
PSS	Progressive Signal System
R-BCAST	Reliable Broadcast
Rand	Random
Repast	Recursive Porous Agent Simulation Toolkit
RL	Reinforcement Learning
RM	Regional Manager
S	System
SA	Simulated Annealing
SCATS	Sydney Coordinated Adaptive Traffic System
SCOOT	Split, Cycle and Offset Optimisation Technique

SL	Supervised Learning
SLA	Service Level Agreement
SOA	Service-oriented Architecture
SOTL	Self-organising Traffic Lights
SPA	Sense-Plan-Act
SR	Success Rate
SS	Survival Space
SuOC	System under Observation and Control
TCP	Transmission Control Protocol
TD	Temporal Difference (Learning)
TLC	Traffic Light Controller
TMC	Traffic Message Channel
TS	Target Space
TTL	Time to Live
TUC	Traffic-responsive Urban Control
UDP	User Datagram Protocol
UTC-NG	Next Generation Urban Traffic Control
VMM	Variable Message Sign
VSM	Viable System Model
WSN	Wireless Sensor Network
XCS	Extended Classifier System

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

During the last decade, research departments of academia and several well-known companies worldwide investigated possibilities to develop new design paradigms for systems that are required to perform high-level management and control tasks in complex dynamic environments (e.g. IBM's *Autonomic Computing* [1]). Examples for the targeted systems include the management and control of vehicular and air traffic systems [2], data communication and telecommunication networks [3, 4], business and production processes, unmanned aerial vehicles [5], or health services [1]. Most of the effort has been driven by the insight that monolithic, static solutions – which make up for the main part of today's developments – are not able to cope with the upcoming demands [6]. Thereby, new and adaptive solutions will become of increasing commercial importance. This insight is accompanied by disappointing experiences in applying conventional design methodologies and techniques to the development of such novel adaptive systems. As a consequence, this insight made way for researchers to focus on a paradigm shift: away from perfectly preplanned monolithic solutions and towards self-organised, distributed, and autonomous entities (cf. [7]).

The high effort allocated for research on new fundamental concepts for self-organised systems is accompanied by current trends. As one popular example, each human of (at least the western) world is equipped with an ever growing number of devices such as handhelds, smart phones, laptops, or music players. Furthermore, these devices are characterised by an increase in their computational power – each of them is as powerful as a standard workstation PC just a few years back in time. Considering just the smart phones – who would have

envisioned that mobile telephones are as small and powerful as they are today only ten years ago? Apart from the basic telecommunication functionality, today's devices are able to perform computer games, applications (e.g. text processing), and handle movie files highly efficiently. They can be used to take part in multi-player games and can be connected to diverse systems via technologies like Bluetooth or WiFi.

Besides the advantages of these technical comforts, some drawbacks can be observed as well. Years ago, nobody would have thought about malfunctions of technical devices due to mutual influences. Nowadays, interconnectedness between devices is not the exceptional case – instead, it is nearly standard. Even if a device is not designed to interact with others, it is subject to their actions in terms of messages, interferences, or just emission and radiation. One aspect of these growing environmental influences is that their impact has to be covered within the design of the system – which leads to more complex solutions that are prone to failures due to this complexity. The manageability of interconnected systems decreases, as users and administrators are just not able to understand and monitor every single aspect of the system. In addition, it is impossible to anticipate all possible situations that a system will be exposed to during its life-time – nobody knows which new trends might appear within the next years or even months [1]. Furthermore, it is hardly possible to explicitly specify the entire behaviour of a complex system on a detailed level, since *complex* is just meant as another word for describing the exorbitant set of theoretically occurring situations and corresponding configurations. From another viewpoint, these systems also become too massive and confusing to be administrable [1]. As one example to underline this observation, *Ganek* and *Corbi* stated that 40 % of today's computer system outages are caused by complexity-induced operator errors [8].

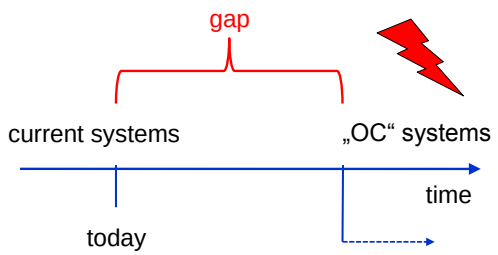


Figure 1.1: Design gap

As a consequence, research initiatives like Organic Computing (OC, [9]), Autonomic Computing [1], or Proactive Computing [10] emerged and proposed to distribute computational intelligence among several (autonomous) entities, since monolithic solutions will not be feasible anymore. The vision behind all of these initiatives is mainly based on designing systems that are characterised by aspects like *local responsibility*, *self-organisation*, *robustness*, *adaptivity*, and *capability of learning*; they differ in the application domain, the degree of the autonomy, and the way to achieve the desired behaviour of the favoured systems. But, to some extent, the previously formulated ideas can be observed among all of them.

Considering the responsibilities of these targeted systems and the mentioned key-characteristics, one fundamental aspect is that they will only achieve the desired behaviour, if they are able to adapt themselves and their behaviour to changing environmental conditions.

This is accompanied by the needed ability to perceive such changes in the first place, and simultaneously control and continuously improve their own behaviour. As a consequence, adaptivity and self-improvement have to define the foundation of such systems. Assuming that it will take time for industry and science to develop satisfying solutions that cover all necessary aspects, a *design-gap* can be identified. Today's systems reach their limits and the desired solutions are not available. Figure 1.1 illustrates this gap. This thesis intervenes at exactly this gap. The basic assumption is that the existing OC principles can already be applied to real-world systems in order to evoke the desired effects. Besides closing the gap, it can be desirable to enable OC characteristics after finishing the design process in order to unburden the designer from one part of the design task. Consequently, the question arises how *existing* systems can be augmented with the desired behaviour. A combination of existing technologies and the advantages of OC's basic principles like adaptivity and self-improvement is needed.

1.2 Problem Statement

"*It is not the question whether adaptive and self-organised systems will emerge, but how they will be designed and controlled.*"

- Prof. Dr. Hartmut Schmeck, Karlsruhe Institute of Technology [7]

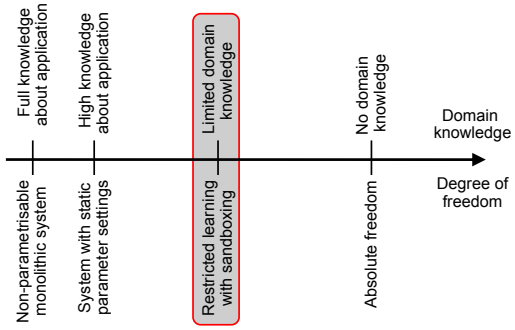


Figure 1.2: Classification of the system's degree of freedom and the corresponding domain knowledge

This thesis presents a mechanism to close the considered design gap by introducing an architectural framework for parametrisable systems. This architecture allows for a self-configuration of the system. It consists of a control mechanism and a parametrisable productive system. This results in an improvement of the adaptation strategy over time without having any insight into the system's logic or domain knowledge. Although the presented approach transfers a certain degree of freedom from the designer to the system itself, this degree of freedom is not unbounded. Figure 1.2 illustrates the possible range. Non-parametrisable systems have a static character and a high degree of integrated domain knowledge – provided by the developer at design time. Parametrisable systems have a slightly higher degree of freedom, since they can be customised to specific environments. In

contrast, systems where the output is obtained by using techniques like Evolutionary Programming have a high degree of freedom, but limited domain knowledge – the evolutionary process needs a feedback about the quality of its found solution in order to perform a guided search, but it decides independently of the underlying problem about its next action. The targeted system constitutes an in-between solution by extending the system's freedom to a certain degree – the developer has to provide some domain knowledge, but the system decides about its actions (here: its configuration) at runtime within given boundaries.

This corresponds to the paradigm shift supported by the OC initiative. Traditional system design follows a top-down path from the system level to the lower implementation levels, while OC wants to move at least a subset of design time decisions to runtime [11]. In today's systems, all future situations affecting the system have to be anticipated and covered at design time. Obviously, this is hardly possible in cases of highly dynamic and non-deterministic environments typically characterising real-world systems. Increasing the system's degree of freedom by moving a set of configuration decisions to the runtime and hence into the system's own area of responsibility should be accompanied by another effect: increasing the robustness in terms of a larger range of acceptable disturbances. This paradigm shift has some fundamental consequences on the design process of such systems:

Self-configuration: The manual configuration task typically performed by searching for a static configuration, which works well in most simulations, is replaced by self-configuration. Classical system design anticipates situations and defines strategies, while OC systems reconfigure themselves at runtime in response to user directives, external stimuli, or disturbances. Therefore, the system needs an active component performing this adaptation task, which has previously been covered by the designer of the system. In the context of OC, systems are assumed to consist of several interconnected subsystems collectively achieving a common goal. Due to the vast set of possible configurations and corresponding situations, the adaptation cannot be assigned to centralised elements working on behalf of the group. Thus, on the one hand each of these interconnected subsystems is responsible for reconfiguring itself, while on the other hand the reconfiguration of the complete system is agreed upon by the subsystems using decentralised negotiation mechanisms.

Self-improvement: In classical system design alternative solutions are discovered at design time, followed by evaluating candidates using simulation or building of real models. Typically, templates or best-practises exist transferring the knowledge from one designer to another. This knowledge can be assigned to the OC system to some degree by the designer, but a large part has to be discovered autonomously at runtime. In the considered case, this knowledge concerns the modification strategies for configuring the system according to changing environments. Thus, the OC system has to be able to explore possible alternatives. In the first place, the selection of predefined possibilities is feasible. But it bears the disadvantage of limiting the design space to a small set of anticipated solutions. In contrast, the system has to be allowed to discover completely new solutions at runtime – which requires the usage of learning and optimisation techniques at runtime. These basic concepts allow

for a self-improvement of the system, but also result in safety and acceptance problems. Hence, the trial parts have to be bounded to guarantee an acceptable system behaviour.

Sandboxing: Already in current system design, model- and simulation-based approaches are standard. These approaches are characterised by the advantage of being able to analyse the system behaviour (e.g. exposing the system to extreme situations) without taking safety-restrictions or user-centric aspects into account. Consequently, a testing within a protected and isolated area without impact on productive parts is done – this area is called *sandbox*. The term can be found in a variety of research areas, but is typically used with the same meaning of the underlying concept – something is tested in an isolated area, where it cannot cause any undesired effects in the real system. Examples include the *sandbox* as a "flexible and expressive thinking environment" [12], to analyse native x86-code [13], and as a dynamic environment in the grid [14].

The same concept has to be transferred from design time to runtime in order to benefit from its advantages. For a large range of applications, a variety of simulation tools and models exist due to the analysis process at design time. These simulators cover a wide range from microscopic (on the lowest level) to abstracted macroscopic (on the highest level) approaches. Using such a sandbox at runtime leads to some basic considerations. Simulation makes only sense if conclusions can be drawn from the results. Accordingly, an optimisation component is needed, which is capable of using the sandbox as evaluation function. Such an optimisation component requires a certain time horizon, meaning the on-line part of the system cannot wait for the new solutions. Additionally, the simulation models have to resemble the environment's actual state as closely as possible – but such a model will probably never reflect an exact copy of the reality (already caused by perception using sensors). Consequently, the present thesis investigates the cooperation of time-delayed optimisation of configuration sets and on-line learning based on these off-line discovered rules.

Application: Investigating learning in OC applications happens not just as an end in itself – instead, the applicability of the developed concepts has to be demonstrated by means of practise-oriented systems. Hence, practise-oriented and problem-related application domains have to be identified and exemplarily investigated. These application examples have to follow two goals. On the one hand, they can be used to demonstrate the benefit of the developed framework in comparison to standard approaches. On the other hand, they shall constitute major contributions to the state of the art in their particular domain by shifting the boundaries of knowledge a bit further. Besides exemplarily investigating application scenarios, a statement about the general applicability of the developed concepts is needed. Hence, one part of the thesis' challenge is to identify similarities between real-world systems, develop a basic classification, and investigate for which of these classes an application of the proposed framework is promising.

1.3 Scientific Focus and Contribution

Based on the previously introduced research problems, the scientific focus of this thesis and the contribution to scientific knowledge can be summarised as follows:

- **Architectural framework:** This thesis presents an architectural framework, which allows for self-adaptation and self-configuration of existing parametrisable systems. Besides the pure adaptation aspect, wrapping existing systems into the framework equips them with further OC characteristics like robustness against a set of disturbances, flexibility, and self-improvement. Thereby, the framework works as a black box solution without interfering with the underlying system's logic.

- **Machine Learning:** The self-improvement is realised using machine learning techniques. Therefore, this thesis characterises the specific learning problem and investigates which techniques are applicable. Additionally, modified variants for the most promising techniques are developed to cover the restrictions and requirements of real-world systems and their safety-demands.

- **Optimisation:** The learning part distinguishes between on-line learning from feedback and a "sandbox" solution to explore new behaviours. Thus, a major focus of the thesis is put on investigating a separation of the learning aspects by encapsulating the exploration part. Therefore, a *sandbox* environment based on simulation has been conceived and the applicability of different optimisation heuristics to the search problem is focused.

- **Traffic Control:** Based on the general design, the thesis presents a new contribution to the state of the art in traffic control systems. Considering realistic models of traffic situations, the advantage of using the novel approach is demonstrated in comparison to existing manually optimised traffic control strategies.

- **Data communication:** Data communication protocols are another application area characterised by systems using mainly static and manually configured solutions. Therefore, a system based on the proposed framework has been developed describing a novel approach to enable situation-aware data communication.

- **Generalisation:** Finally, the applicability of the developed approach has been investigated by generalising the control problem. Considering this generalisation and abstraction, the question is answered for which type of systems an application of the framework is promising and where restrictions might be expected.

1.4 Outline

This thesis is structured as follows. Chapter 2 gives an overview of related work that has been published in the domain of architectures for adaptive systems. Apart from related solutions, the generic Observer/Controller architecture is explained, which serves as a basis for the presented work. Afterwards, the proposed system is introduced on an architectural level in Chapter 3, accompanied by formally defining its scope and target. The architecture describes a general concept, but leaves some design choices to the developer. In particular, on-line learning and a safety-based off-line optimisation component are used – Chapter 4 discusses the characteristics of these components and determines which specific techniques should be used to cover the corresponding tasks. Chapter 5 describes the structure of the evaluation. Afterwards, the following three chapters demonstrate the applicability and the potential benefit of the approach by applying it to different application domains: Chapter 6 introduces an adaptive traffic control system for urban road networks. Chapter 7 presents a system to dynamically adapt data communication protocols to changes in their environmental conditions. Chapter 8.1 discusses how the developed solution can be applied to adaptive production control scenarios, while Chapter 8.2 demonstrates the applicability to error prediction in enterprise mainframes. These four different applications are based on the generic approach as discussed in the previous parts of the thesis – but they are not intended to just serve as exemplary application scenarios. Instead, the evaluation part for all of them demonstrates the unique characteristics of the solutions and the advantages compared to the state of the art in their particular domains. In Chapter 8.3, the scope of the framework is investigated by applying it to an abstracted mathematical problem. Finally, Chapter 9 contains the conclusion of this work and gives an overview of promising future research opportunities.

1.5 How to Read this Thesis

This thesis contains several different aspects of diverse research domains. Of course, it is conceived as a whole and intended for reading it in one piece. But – due to the variety of covered areas – some readers might only be interested in partial aspects. Hence, the following part aims at guiding readers to those areas they are mostly interested in. Consequently, the guide is organised along the particular research areas covered by this thesis.

System Design: The basic idea of this thesis is to develop the system's design and a corresponding framework to enable self-configuration and self-improvement for parametrisable systems. Hence, a major part focuses on system design aspects. Readers, which are only concerned with this issue, will read Chapter 2 for a discussion of related work and Chapter 3 for insights on the system developed in this thesis.

Machine Learning: The aforementioned aspect of enabling self-improvement capabil-

ities to fine-tune the self-configuration of adaptive systems is achieved by using machine learning techniques. In this context, a special focus is set on learning under safety restrictions. Researcher concerned with machine learning topics will find their major field of interest covered by Chapter 4.1. This chapter discusses the applicability of machine learning techniques to the control problem defined in the framework. Afterwards, two novel variants for rule-based on-line learning mechanisms are introduced: a modified Learning Classifier System and a modified Fuzzy Classifier System.

Optimisation Heuristics: Besides on-line learning capabilities, the framework contains a component to discover the best possible response for currently unknown situations. Finding the best response is typically referred to as an *optimisation task* – hence, one part of this thesis deals with choosing appropriate search heuristics for the corresponding optimisation problem. Researchers focusing on this topic will find information regarding their research area in Chapter 4.2.

Traffic Engineering: Researchers predominately interested in traffic engineering topics will focus on Chapter 6. The chapter describes the Organic Traffic Control system as one application for the framework presented in this thesis. Hence, the chapter provides a comprehensive analysis of decentralised traffic control including the state of the art, the customisation of the developed framework, and an extensive evaluation of real-world scenarios.

Data Communication: A second major scope of applying the framework is set on data communication networks. Therefore, Chapter 7 introduces the Organic Network Control system, which adapts parameter configurations of data communication protocols automatically to changing environmental conditions. Analogously to the chapter concerned with traffic engineering aspects, Chapter 7 provides a comprehensive analysis of network protocol parameter control including the state of the art, the customisation of the developed framework, and an extensive evaluation using three different protocol types.

Chapter 2

State of the Art

Enabling adaptivity at system level has been investigated by researchers for years. As a result, several architectures or design patterns have been presented carrying attributes like *adaptive*, *self-adaptive*, or *dynamic response*. Further systems focus on more locality-based principles like self-organisation, self-configuration, or self-management. All of these concepts are by no means completely new. Thus, this chapter intends to give an overview of existing related approaches and to highlight the need of a novel system, which is able to cover the research questions discussed in the previous chapter.

The remainder of this chapter is organised as follows. Initially, Section 2.1 defines requirements for adaptive systems covering the problem statement of the previous chapter. Afterwards, research areas investigating adaptive systems are considered following a systematic approach:

- Initially, the particular domain and the relation to the topic of this thesis is motivated.
- Afterwards, basic architectures and design principles to achieve adaptivity in this research area are presented.
- The third step investigates to which extent these basic principles have been transferred to applications and prototypical implementations.
- Finally, these three aspects are used to characterise the research domain with respect to the initial requirements and analyse their applicability to the identified problem.

Thereby, the Observer/Controller design pattern as developed in Organic Computing (OC) [15] is set apart from other work, since it will serve as input and basis for the developed framework.

In the remainder of this thesis, the terms “design”, “architecture”, “design pattern”, and “framework” are used as follows. *Design* and *architecture* describe a general macrostructure

of systems or a systematic concept of how to construct systems, respectively. A *design pattern* is a concretisation of the former two abstract system descriptions. It decomposes abstract tasks into more specific subtasks and describes the cooperation of the resulting elements. The following discussion of the OC domain in Section 2.2 serves as example: Figure 2.1 describes a general architecture, while Figure 2.2 depicts a design pattern, which is based upon the general architecture. In contrast, the term *framework* denotes a concrete implementation of a group of systems that are all following the same methodology and using the same techniques – right up to a collection of class libraries and interfaces. Following this classification, there is only one class below the framework: the concrete *system* itself, which is serving a specific purpose.

2.1 Classification of Requirements

Chapter 1.3 introduced the scope of this thesis and defined the general requirements for the system to be developed. In order to analyse the state of the art in more detail, the necessary classification of these requirements is introduced in the following part. The identified aspects are then used to characterise the particular concepts: either they can deal with the identified problem or the demand of a new solution exists. Consequently, the following aspects define requirements that have to be fulfilled as completely as possible. Therefore, they describe attributes of an additional control mechanism that can be applied to allow for the desired characteristics as introduced in the previous chapter.

A) Adaptivity: As initially defined, it is assumed that augmenting productive systems with the additional ability to adapt themselves to changing environmental conditions will lead to a higher system performance. Thus, an external control mechanism is needed that allows for adapting the system in response to observed changes of attributes with impact on the system's performance. The result of this adaptation process is an increased performance in terms of problem-specific metrics of the productive system's particular domain.

B) Robustness: Closely connected to the goal of achieving adaptivity is the need of robustness against a set of disturbances in the sense of [16]. Changing parameter configurations according to observations is assumed to increase the system's performance. In this context, disturbances are not only failures and misbehaviour of components, but also unexpected situations where the static setup of parameters leads to a dissatisfying performance. In contrast, changing them as response to disturbances and consequently keeping the system's behaviour within tolerated boundaries is an even more challenging task. As a result, the need of users' manual intervention (and the associated cost) can be drastically decreased.

C) No interference with the system's logic: The previously demanded adaptation has to be reached without interfering with the system's logic – an additional control mechanism surrounding the system has to provide a customisable black box solution and

to work on the available set of accessible configuration parameters. An intervention to the logic of the system instead of configuring accessible parameters would entail that the control mechanism cannot be realised as black box solution anymore. Consequently, this requires the existence of such parameters for the system to be controlled.

D) Operability: The additional control mechanism is not allowed to affect the productive system's operability. In cases where the additional adaptation mechanism fails (e.g. due to malfunctions of components), the underlying productive system must be able to continue its work (**remain operable**) – merely with a static character.

E) Flexibility: Current systems are characterised by processing static logic or at least following a predefined goal. In future systems, the aspect of being able to change this goal on demand will become of increasing importance. Thus, the control mechanism has to provide adequate concepts to incorporate flexible goals and to change them at runtime.

F) Vast situation and configuration spaces: Based on the initial motivation of OC [17], it is assumed that systems have to cope with vast situation and configuration spaces. Closely related to the corresponding indefinite possibilities for observations and configurations is the need of coping with unknown and unanticipated situations. The control mechanism has to find appropriate settings, although it cannot fall back to predefined strategies and does not know the optimal response.

G) Self-improvement: Adaptivity can be achieved by different approaches ranging from choosing between a restricted set of predefined behaviours to allowing the system to explore new behaviours autonomously. Due to the assumed vast situation and configuration spaces characterising real-world systems, a predefinition of alternatives is assumed to be inappropriate. Thus, the control mechanism has to be capable of self-improving its behaviour by taking a feedback on its actions into account. As a result, appropriate techniques to perform self-optimisation and self-improvement are needed.

H) Restricted exploration: Self-improvement relies on choosing between alternative actions and autonomously identifying new behaviours in case of unanticipated situations. In addition, a qualitative feedback is needed distinguishing between good and bad behaviour. Although this concept has to make use of exploration mechanisms, real-world systems require that only tested and acceptable actions are performed – otherwise it cannot be guaranteed that the system will always behave in an acceptable manner. Thus, the system has to be equipped with an effective mechanism to restrict the exploration parts of self-improvement – *only pretested solutions are allowed*. Since such a pretesting directly before applying the action to the system is infeasible at design time due to the vast situation spaces, a "sandbox" solution is needed.

I) Decentralised operation and collaboration: OC assumes that a set of autonomous, more simple systems will replace the existing, monolithic ones to counter the system's complexity [7]. Since a controlled self-organised behaviour of several cooperating elements is desired, the system architecture has to foresee decentralised collaboration possibilities. Although collaboration is typically application- and task-specific, the possibility

has to be covered by the architecture by means of e.g. communication capabilities.

J) Comprehensibility: In order to achieve user acceptance, the control mechanism has to provide appropriate interfaces for monitoring. In addition, the actions performed by the adaptation component have to be comprehensible to users. An engineer analysing the past behaviours by comparing input situations and applied actions has to be able to understand why the system acted in the particular way.

K) Real-world requirements: OC and related research initiatives develop novel concepts for systems applied to the real world. Thus, the control mechanism enabling adaptivity for the productive system has to be able to deal with environments that are typically noisy. The term "noisy" summarises various influences such as measurement errors, incomplete observations, transmission errors, or continuous values that have to be treated like random influences.

L) Generalised approach: In addition to these criteria A to K, a generalised framework is needed, which does not provide just a domain-specific solution. In particular, it has to allow for enabling adaptivity for a variety of systems and tasks.

Based on the aforementioned aspects (the previous itemisation from A to L) that are characterising the requirements of the desired adaptation module (the control mechanism), a comparison to the state of the art is possible. Hence, the remainder of this chapter identifies related research areas and discusses existing concepts to cover similar or connected problems. Therefore, the ordering of the aspects will be kept for each considered technique from the state of the art.

2.2 Organic Computing

This thesis falls into the context of OC [17] – hence, it seems natural to start the search for appropriate and suitable architectures, design patterns, or frameworks in this research field. OC has emerged recently as a challenging vision for future information processing systems [7] and claims that a paradigm shift in system development is needed. Comparable to the "Vision of Autonomic Computing" [1], which predicts that the increasing interconnectedness of systems and devices will become a "nightmare" for designers and administrators of IT infrastructure, OC postulates the need of a *paradigm shift* in systems engineering towards self-organised solutions. Nowadays, more and more systems are equipped with sensors and actuators, aware of their environment, and communicating freely. Based on these characteristics, future OC systems will be able to self-manage their behaviour, and a collection of these systems will be able to self-organise to cooperatively achieve tasks. However, the designer of the system will be able to delegate control to populations of smaller, more autonomous, and collaborating entities. In this context, *autonomous* means that the particular system is able to work in its environment without external control.

In contrast to domain-specific initiatives like Autonomic Computing for the IT infrastructure environment, OC covers a broader spectrum of systems. Thus, current research investigates heterogeneous applications and theoretical concepts ranging from hardware [18, 19] over robotics [20, 21] to software [22]. An overview of the concepts and basic ideas of OC is given in [23, 24]. Although the focus is broad, some common ideas, especially for the basic design of systems, can be observed. The most prominent example in system design is the "Observer/Controller design pattern", which has been introduced in [17] and further refined in [25, 26]. A summary of the developed approach and its applications is given in [15].

2.2.1 The Observer/Controller Design Pattern

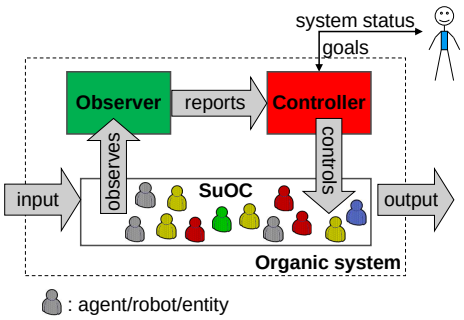


Figure 2.1: Basic concept of the Observer/Controller architecture [26]

OC systems are characterised by the need of an adequate response to environmental or internal changes. Typically, this response results in an adaptive behaviour and incorporates further aspects like *robustness* and *flexibility*. In order to allow for such an adaptation process, the system's design provides a regulatory feedback mechanism capable of monitoring, analysing, and reacting to changing conditions. Therefore, OC proposes the so-called *generic Observer/Controller* design pattern, which constitutes a generalised way to achieve controlled self-organisation in technical systems [26, 25]. This regulatory feedback mechanism contains three major components (see Figure 2.1 and Figure 2.2 for a more detailed version):

System under Observation and Control (SuOC) The SuOC is the "productive" part of the system that serves a specific purpose. The term corresponds to the previous notation of the "existing parametrisable system", which has to be controlled by the framework. Thus, the SuOC is functional without observer and controller and it will remain operable if higher layers fail (i.e. *Observer/Controller* components).

Observer The SuOC's state and dynamics are monitored by the observer in order to give an appropriate description of the current situation for the whole system at each point of time. The observer also monitors the environment, either directly or through the sensors of the SuOC.

Controller Based on the observer's aggregated information, the controller influences the SuOC with respect to the goals given by the user.

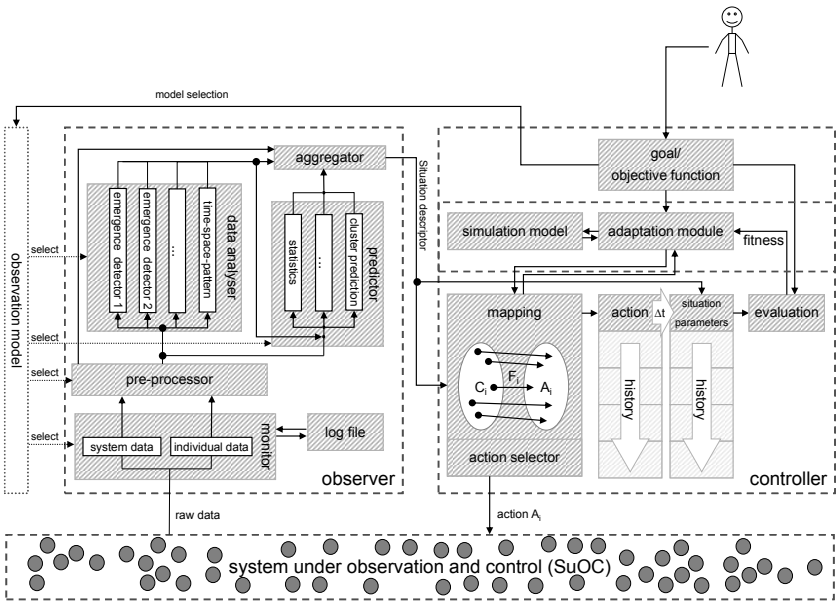


Figure 2.2: Generic Observer/Controller design pattern [26]

System under Observation and Control: The lowest layer of the architecture encapsulates the productive part of the system. This productive system can serve various purposes, see Section 2.2.3. Higher layers of the architecture monitor and adjust (if necessary) the parameter configurations of the productive system in discrete time intervals. OC postulates the distribution of computational intelligence among large populations of smaller entities – thus, the SuOC in Figure 2.2 can refer to single systems or groups of autonomous systems, respectively. In both cases, the SuOC needs to fulfil some basic, application-specific requirements:

- The SuOC's behaviour and its environmental conditions have to be observable.
- The performance of the SuOC according to some goal given by the designer or user has to be measurable.
- The SuOC has to possess a set of variable parameters that can be dynamically adapted at runtime and that have certain impact on the performance of the system.

Observer: The observation task can be split into five consecutive steps: *monitoring*, *preprocessing*, *data analysis*, *prediction*, and *aggregation*. The monitoring part receives the raw data from the SuOC and is based on an observation model that customises the observer's functionality for the specific SuOC. Thus, it selects the observable attributes of the system, the analysis detectors, and the appropriate prediction methods. This selection is done within constant discrete time intervals, expressed as *sampling rate*. The observed system data can consist of the SuOC's individual data and some additional more global

(environmental) system attributes. All measured data is stored in a log file for every loop of observing/controlling the SuOC, since it has to be accessed by the predictor and by the data analyser (see Figure 2.2), e.g. for calculating time-space patterns.

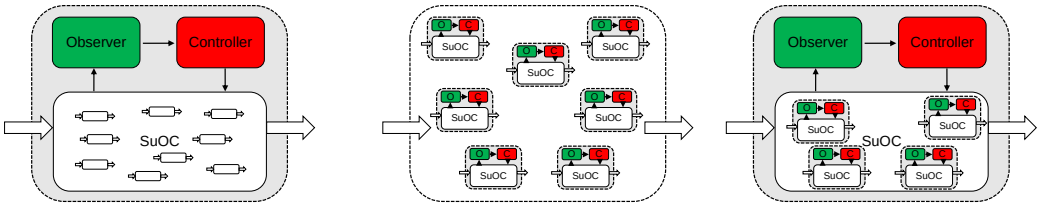
The actual observation process starts with receiving the SuOC's raw data. This raw data can be preprocessed in order to smooth the corresponding attributes and to extract meaningful attributes – afterwards, it is transformed into a vector describing the system's *situation*. Next, the data analyser applies a set of detectors to this preprocessed data vector. For instance, cluster computation, detection of emergence parameters (according to the definition in [27]), or further mathematical and statistical methods might be needed to extract meaningful information. The result of this step is a system-wide description of the SuOC's current state. Since the adjustment of the SuOC's parameters is performed according to the sampling rate, the controller's action has impact on the next situation. Thus, the attributes from the situation vector can be augmented with forecasts for both – the next raw data as well as the next system-indicators (by using specific or statistical methods like chart analysis). Finally, the aggregator collects the processed information into the so-called *situation descriptor*, and passes it to the controller, which appropriately influences the SuOC. Further details on the observer part with a special focus on determining emergent behaviour have been discussed by *Mnif* in [28].

Controller: The controller's task is to guide and control the SuOC by choosing its most promising parameter configuration. It receives the processed and augmented situation descriptor from the observer and interferes only in those cases with the SuOC where an adaptation of the current settings is necessary. The decision module that is responsible for choosing the most appropriate parameter settings for the current situation is called *action selector*. Since OC systems act in real-world environments, a fast decision is necessary. Thus, the action selector is equipped with two important capabilities: on-line learning and preparation. The learning part works on a fixed set of different strategies that map a given situation onto a corresponding action – it aims at increasing the quality of this selection process. The task is performed in real-time. In addition, the preparation component is responsible for extending the behavioural repertoire of the action selector and has a broader time horizon. A detailed investigation of the controller part with a special focus on the learning component has been performed by *Richter* in [29].

As a conclusion, it is important to note that an organic system will continue to work, although observer and controller might be disturbed and stop working. Thus, the main objective of the proposed architecture is to achieve a controlled self-organised system behaviour. In comparison to classical system design, OC systems have the ability to adapt and to cope with some emergent behaviour they have not been programmed for explicitly.

2.2.2 Design Variants of the Observer/Controller Design Pattern

The generic Observer/Contoller architecture needs to be customised to different scenarios by adapting the various components of the observer and the controller. As stated in [30] and depicted in Figure 2.3, this customisation of design variants ranges from *fully central* to *fully distributed* architectures. The former case describes a single Observer/Controller that regulates various components of the SuOC and that directly intervenes into all of these entities (see Figure 2.3(a)). In contrast, the latter example defines one Observer/Controller for each component of a technical system (see Figure 2.3(b)). These two variants – the fully central and the fully distributed architecture – define the two extreme points in the design space. Nevertheless, there are also many other distribution possibilities like a multi-level architecture (see Figure 2.3(c)).



(a) Central (b) Distributed (c) Multi-level

Figure 2.3: Design variants of the generic Observer/Controller design pattern [30]

Based on these various possibilities to realise and customise the generic Observer/Controller architecture, the designer of the system has to decide about the most promising approach for his context. In the course of this decision process, the need for different design variants can be classified according to increasing size, complexity, and heterogeneity of the contained subsystems. The simplest case is an isolated system with a clearly defined purpose and a restricted configuration space where no distribution is needed (see Figure 2.1). In contrast, larger and more complex systems are characterised by a drastic increase of the number of possibly occurring situations and the corresponding number of different system configurations that cannot be handled by one single Observer/Controller component. With growing complexity, the demand of a hierarchical and multi-levelled decomposition of the control problem becomes more recommendable. A common analogy for such systems is the organisational structure of large companies. Some kind of management serves as highest instance that defines abstract global goals or strategies. Lower layers of the hierarchy convert the abstract goals for their area of responsibility into more specific goals – hence, high level administration units are not bothered by low level decisions [29].

OC introduces the variability of systems as a measurement for the quantification of complexity during the design process for technical systems. The term *variability* is defined as the number of possible configurations of a SuOC [31]. Obviously, the variability tends to increase with the complexity of the SuOC. However, introducing hierarchical and multi-

levelled Observer/Controller structures is a powerful instrument to reduce the externally visible variability and consequently hide the complexity of a system.

2.2.3 Application of the Observer/Controller Concept

The generic architectural design and its distribution variants describe an architectural blueprint. Besides the application scenarios developed in the context of this thesis, several projects from varying application domains have developed their own systems that are built upon this basic concept or are at least inspired by it. This section provides a survey of various applications. The survey is structured according to the distribution of the Observer/Controller components. An overview and a detailed discussion of these projects has been presented in [15] and is summarised in the remainder of this section.

Central Observer/Controller

Technical systems that cannot be subdivided into subsystems or that consist of highly interrelated subsystems are monitored and controlled by a centralised Observer/Controller architecture, see Figure 2.3(a). A first example for such an application is the **organic elevator control** system as presented in [32]. Elevators are typically working according to a simple mechanism – they stop at the nearest hall call in their current running direction and only change their direction after serving all requests for the current one. In case of a group of elevators working in parallel according to this simple concept, a synchronisation effect can be observed when all elevators move up and down as a parallel wave. This so-called *bunching effect* results in increasing waiting times for passengers and has been proven as inefficient [33]. In terms of OC, this is an *undesired emergent effect* that has to be detected and avoided. Therefore, the organic elevator control system consists of several autonomous elevators and one centralised Observer/Controller component. The observer contains corresponding emergence detectors, and the controller can intervene to discontinue the synchronisation by manipulating the behaviour of individual elevator cabins.

The project **Organic Computing in Off-highway Machines** (OCOM) focuses on machine management in off-highway machines like tractors or wheel loaders [34] and serves as second example for a centralised design variant. Each off-highway machine consists of several subsystems (like the traction drive, the power take-off, and various auxiliary components) that are closely interrelated. Hence, an efficient operation of the machine (e.g. in terms of a minimal fuel consumption) can be achieved by developing an adaptive machine-wide management of these subsystems. Therefore, OCOM relies on a centralised Observer/Controller design. The SuOC is formed by the machines' subsystems, while the Observer/Controller is responsible for a reliable, adaptive, and robust machine management.

A third example for centralised Observer/Controller systems are self-organised **cleaning**

robots [29]. These robots search in their local neighbourhood for dirty places and clean them by following a local strategy. Additionally, they try to self-improve their behaviour by *learning* using the success of the cleaning strategy as evaluation criterion. They are able to indirectly communicate with each other by placing "pheromones" at places they have already cleaned – these pheromones are observed by other robots. The goal of this communication is to avoid double-cleaning of areas and the resulting wasting of resources. Therefore, a centralised and a distributed variant have been investigated – the former one uses a centralised component that generates and exchanges the robots' behaviour strategies [35], while the latter one analyses fully decentralised learning strategies of the autonomous robots [36].

Distributed Observer/Controller Components

As depicted in Figure 2.3(b), a second design variant for OC systems is useful if a technical system consists of several loosely coupled subsystems. In these cases, each subsystem can be equipped with a separate Observer/Controller component. The predominant application area of this type are applications where the subsystems are locally distributed or belong to different authorities. A prominent example are Service-oriented Architectures (SOAs) that typically consist of several distributed applications and are characterised by a high degree of interaction among these components. Based on the general concept, the design of **Organic Service-oriented Architectures** (OSOAs) [37] has been investigated where management responsibility at runtime is distributed to each component. Therefore, each SOA entity is equipped with an Observer/Controller component to achieve controlled self-organisation [38]. The observer monitors its SOA component and determines the current operational state. Based on this information, the controller adapts the behaviour of the underlying SOA component according to given objectives specified in service-level agreements (SLA). Furthermore, automatically negotiated SLAs are used to coordinate the runtime behaviour of service components according to given business objectives [39].

Multi-levelled Observer/Controller Components

The third type of OC systems is designed according to the variant depicted in Figure 2.3(c). A set of distributed Observer/Controller components is hierarchically organised with those on a higher level influencing subsystems on lower levels. Such a design can be found in systems where several subsystems are sufficiently complex to require their own Observer/Controller. One example for such a system has been developed in the context of the project *MeRegioMobil* that investigates a smart home environment equipped with several household appliances and an electric vehicle [40]. Within this smart home environment, charging periods of the vehicle and the operation of various appliances (like freezer or washing machine) are automatically rescheduled in order to adapt the consumers' energy demand

to the power generation in the grid. This rescheduling is done by taking price signals into account, which reflect a load-prediction of the energy grid, and considering constraints given by the user. Therefore, a multi-levelled Observer/Controller framework is used. Each appliance is equipped with a local Observer/Controller component responsible for observing the appliance's current state and turning it on or off according to current conditions. Additionally, these local Observer/Controller components communicate their data (e.g. power consumption profiles) to a higher-level smart home management device that centrally derives timing strategies for the smart home.

2.2.4 Characterisation of the Observer/Controller Design Pattern

Considering the list of requirements as introduced in Section 2.1, the *generic Observer/Controller design pattern* can be characterised as follows:

A) Adaptivity: By dynamically adapting the SuOC's parameters, the general design of OC systems aims at **enabling adaptivity**.

B) Robustness: The approach is designed to provide **robustness against disturbances**, in particular those induced by emergent phenomena.

C) No interference with the system's logic: Although the main focus of the concept is to control emergent behaviour [27], the application of parametrisable systems **without interfering with their logic** is possible.

D) Operability: Due to the clear separation and the non-intrusive concept, the SuOC **remains operable** (with static configurations) in cases where the Observer/Controller component fails (the *principle of non-critical complexity* [11]).

E) Flexibility: The design pattern includes an interface for the user to define the system's goal. Thus, the design covers **flexible goals**. However, this feature has not been investigated at a technical level, yet.

F) Vast situation and configuration spaces: The generic Observer/Controller design pattern aims at dealing with **vast situation and configuration spaces**. The particular realisation of observer and controller part decides whether **unanticipated situations** are managed appropriately or not.

G) Self-improvement: The controller part of the concept contains a rule-based *action selector* and an evaluation approach. Although not explicitly named in Figure 2.2, the corresponding tasks can be covered by a **self-improvement** component.

H) Restricted exploration: The controller part contains a simulation model, which can be used to **restrict the exploration mechanism** of the learning component – it defines an abstract concept rather than a detailed classification of tasks.

I) Decentralised operation and collaboration: The design pattern does not include provisions for a **decentralised collaboration**. However, extensions are possible and subject to ongoing investigations [11].

J) Comprehensibility: Due to a rule-based approach, the behaviour of the system is **comprehensible to users** – but this depends on the realisation of the learning part and the freedom granted to this component.

K) Real-world requirements: The architectural concept has been designed to deal with **real-world requirements** – in particular, continuous and noisy sensor information can be used and disturbances are covered by the adaptation component.

L) Generalised approach: The Observer/Controller design pattern proposes a generalised approach to allow for adaptivity in technical systems, but it lacks steps to substantiate the abstract ideas.

In general, the generic Observer/Controller architecture postulates a design paradigm, rather than defining a detailed concept and an applicable framework. Currently, the architectural design and its components are not completely investigated. For instance, the project *Observation and Control of Collaborative Systems* (OCCS) [41] covers certain aspects only, like the observation model of the observer or the quantification of robustness and flexibility. Only scenario-specific prototype implementations are available, rather than a reference implementation. Furthermore, a major focus has been set on determining emergent behaviour [27, 28, 29] – other detectors are mostly neglected. Due to the abstract character of the concept, it does not provide a black box solution and therefore has to be customised by defining distribution variants, models, and components like the action selector.

Summarising, the Observer/Controller design pattern tackles a large set of the initial requirements. Due to its generic approach and the abstract design, it leaves some questions open that are covered by this thesis: how can restricted and safety-oriented self-improvement be enabled, how can collaboration and rule generation be realised? Furthermore, the way towards an adjustable black box solution is pursued. Thus, the generic concept serves as input and basic model for the framework as developed systematically in this thesis.

2.3 Related Architectural Approaches

Obviously, OC is not the only community addressing complexity by introducing controlled self-organisation and learning or developing adaptive, flexible, and robust systems. Several other software and hardware domains have to operate systems autonomously in unknown environments – again, a certain degree of freedom needs to be granted to these systems while they are simultaneously kept within controlled boundaries. A related survey of design methodologies compared to OC's generic Observer/Controller pattern is given by *Richter* in [29].

2.3.1 Control Theory

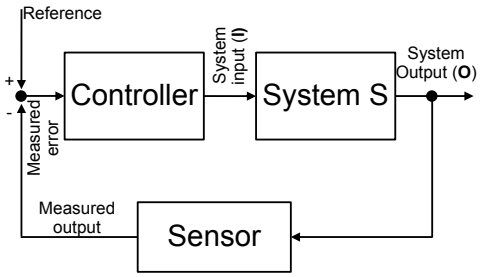
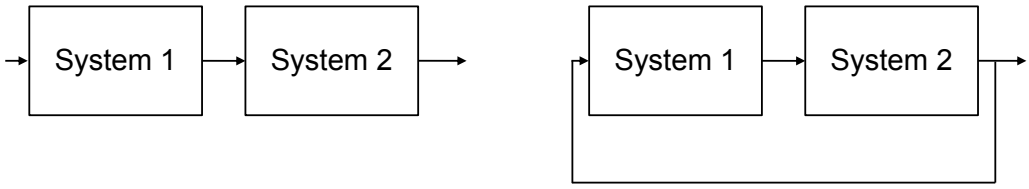


Figure 2.4: Control loop (detailed)

Control theory is an interdisciplinary field of mathematics and engineering and relevant for the control of various physical processes [42]. Some principles can already be found in antiquity, but it has not been established as a distinct discipline until the late 1950s. Based on a mathematical background, it defines control loops to cope with the behaviour of dynamics in technical systems [42]. In this context, a *dynamic* system means that its behaviour changes over time – mostly in response to external stimuli or forces [43]. In the simplest case, a system S produces some kind of output O for an input I. This output O is continuously compared to a predefined reference value R. If O deviates from R, a controller will automatically adapt the input value I in order to satisfy the goal $O = R$. Figure 2.4 illustrates this concept by describing a feedback loop.

Typically, *open* and *closed loop* systems are distinguished. A system is called *closed loop system* if its subsystems S_1 and S_2 are interconnected as a cycle (see Figure 2.5(b)). Non-cyclic variants are referred to as *open loop systems* (see Figure 2.5(a)). The main purpose of each control system is to guarantee the stability of the control loop's behaviour. In case of a linear system, this can be achieved by defining lower and upper boundaries. In contrast, specific theories are needed for non-linear systems. Thus, the control model and the control strategy are responsible for providing the desired behaviour. Three main categories of control models are known in literature: *adaptive control, proportional-integral-derivative control and model-predictive control* [44].



(a) Open loop (b) Closed loop

Figure 2.5: Open and closed loops in control theory

Adaptive Control

In order to cope with time-varying and disturbed sensor data, *adaptive control* [45] modifies the control model used by the controller. A frequently discussed example to motivate the concept is the control of an aeroplane whose mass decreases due to fuel consumption. Thus,

the controller needs a control model that adapts itself according to these changes. The main subject of adaptive control is to dynamically change the respective control law in that sense that it does not need a-priori information about the bounds on these uncertain or time-varying parameters.

Proportional-Integral-Derivative Control

The *Proportional-Integral-Derivative* (PID) controller is the most commonly used variant of a feedback controller. It is the most promising approach in the absence of knowledge about the underlying process [46]. It defines a generalised feedback mechanism that can be found in a wide range of industrial installations, although the particular organisation is application-specific. The concept is based on calculating the deviation of the desired behaviour by comparing the measured value to the desired one – this difference is going to be minimised by adjusting the process control inputs. The calculation is based on three separate parameters: the *proportional*, the *integral* and the *derivative* values, denoted P, I, and D. P is used to determine the reaction to the current error, I refers to the sum of recent errors, and D relies on the rate at which the error has been changing. The process is adjusted using the weighted sum of P, I, and D [47].

Model Predictive Control

One major problem in control theory is to guarantee stability for closed-loop control systems. Introduced in the 1980s, *Model Predictive Control* (MPC) [48] has been successfully applied to industrial installations such as oil refineries and chemical factories. Today, it is one of the most widely used control techniques in process control. A driving car serves as prominent example. Within a given control horizon the driver knows the desired trajectory. Based on the characteristics of the particular car, he decides on the best control actions (accelerator, braking, steering) in order to follow this trajectory. Thus, only the first control actions are chosen at each instant – this procedure is followed repeatedly. In contrast, the decision process is based on previous errors [48] in other strategies like PID.

The example illustrates that MPC controllers are based on dynamic models of the process. The controller itself consists of a multi-variable control algorithm containing a dynamic model of the process, knowledge about past control actions, and an optimisation cost function for the considered prediction horizon. Based on the model and the current measurements, the MPC controller calculates the next actions for the independent variables of the process. The operational part takes independent and dependent variable constraints into account; afterwards, the determined set of actions is transferred to the corresponding desired value of the regulatory controller. Most of the installations are based on linear models (they are approximately linear within a restricted prediction horizon [49, 50]), but also non-linear models are known [51].

Application of the Concept

Applications of control loops and their design variants are manifold and can be found in different installations. Especially *modelling* is ubiquitous in science and engineering with a strong connection to applied mathematics [44]. Thus, most of the literature about control theory includes parts concerned with models using differential and difference equations [52]. The most prominent examples come from physical systems, especially mechanical, electrical, and thermofluid systems [53]. Corresponding representatives of the problems are given as follows. *Arnold* discusses mathematical methods applied to control problems of classical mechanics [54], *Kundur* describes methods and control strategies for power system stability [55], and fundamentals of fluid power control are given by *Watson* [56]. An overview of further examples has been presented by *Aström and Murray* [44].

Characterisation of Control Theory

Considering the list of requirements as introduced in Section 2.1, *control theory* concepts can be characterised as follows:

A) Adaptivity: The approach **enables adaptivity** within the given boundaries provided by the logic of the control model.

B) Robustness: Control theory provides some kind of **robustness against disturbances** if these disturbances do not affect the controller or the functional process of the system and have been considered by the designer.

C) No interference with the system's logic: If appropriate models of the system's behaviour exist and the dependencies of parameter and system performance can be defined, control theory might be adopted to deal with existing systems **without interfering with their logic**.

D) Operability: Due to the strong interconnection between control mechanism and element under control, the underlying process might **not remain operable** in cases where the control mechanism fails.

E) Flexibility: The goal of the control loop is hard-coded – thus, it cannot handle **flexible goals**.

F) Vast situation and configuration spaces: Vast situation and configuration spaces have to be covered at design time by taking them into account when developing the control model. **Unanticipated situations** are typically problematic and might lead to failures.

G) Self-improvement: Current approaches do not consider **self-improvement**.

H) Restricted exploration: Due to the predefined logic, the system does not rely on **exploration mechanisms** to cover unanticipated situations and therefore does not need concepts to control the impact of such a mechanism.

I) Decentralised operation and collaboration: Although systems can be interconnected, **collaboration** to achieve goals cooperatively is not part of the design.

J) Comprehensibility: The **comprehensibility to users** depends on the complexity of the system. Especially in large collections of integrated subsystems with interdependencies, a comprehensibility is rarely given.

K) Real-world requirements: Control theory has been successfully applied to manifold real-world applications – thus, the concepts are able to deal with **real-world requirements**.

L) Generalised approach: Due to the strong relation to the particular control problem, a control loop is always problem-specific. Nevertheless, the approach itself provides a generalised concept.

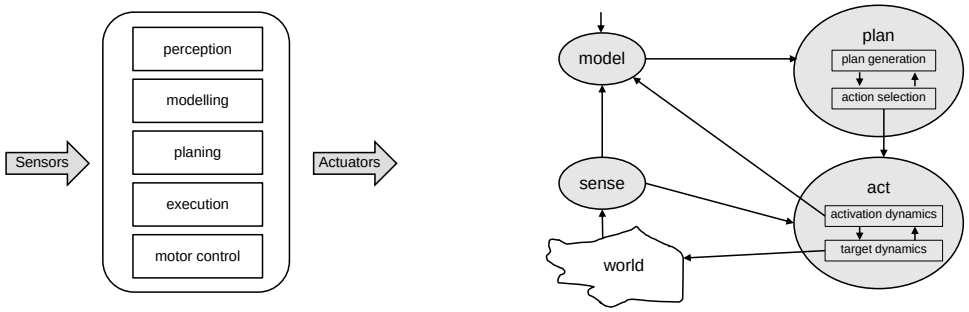
Thus, control theory approaches do not match all the requirements for this thesis, since especially unanticipated situations and the corresponding situation spaces are not applicable. Consequently, the designer has to foresee all possible situations that might appear at runtime.

2.3.2 Adaptive Architectures for Robotics

First of all, probably *robot systems* will come to mind when considering autonomous and environmental-aware systems. Robots are designed to act under real-world conditions without external control of human users. Thus, the robots have to cope with similar problems as investigated in this thesis. Such a *robot* is defined as an entity consisting of principles from system design and kinematics by combining sensors, actuators, and information processing within one autonomous system [57].

Architectural Concepts

Several different approaches to design robot systems have been proposed and investigated during the past decades [57]. Those being most closely connected to the topic of this thesis are considered in the following. *Sense-Plan-Act* (SPA) was the predominant robot control methodology until mid of the 1980s [58]. It defines a simple control loop consisting of three phases. Phase one is used to gather all available information from the sensors (sense), in phase two a world model is built upon the gathered information and the next move is derived (plan). Finally, this plan is executed (act) in phase three. Figure 2.6(a) illustrates the approach and shows that the planning phase distinguishes between five consecutive steps: perception, modelling, planning, task execution, and motor control. During these steps, a model-based determination of the next steps is performed and an appropriate control strategy for the motors is calculated. SPA is used in iterations, which means that after finishing the planning phase the next sensing phase starts immediately.



(a) Sense-Plan-Act [58] (b) Model-based extension [59]

Figure 2.6: The Sense-Plan-Act paradigm

Using the terms of control theory as discussed before, SPA is designed as an *open control loop* (see Figure 2.5(a)) – the information flow through all three phases is linear from sensing over modelling and planning to acting. Thus, the design and the corresponding behaviour are easy to understand. But especially for the restricted computational capabilities of mobile robots, the computationally intensive approach has drawbacks. A first approach to cope with these drawbacks has been a separation of reactive and planning behaviour as proposed by *Herzberg et al.* [59], see Figure 2.6(b). An alternative approach spearheaded by *Brooks* [60] became more prominent. It decomposes the robot control problem into sub-problems – these smaller problems are task-specific modules (Brooks calls them "behaviours"). They are realised as a network of finite state machines. This *subsumption architecture* distinguishes between several predefined behaviours. As a result, the planning part of choosing the best behaviour or strategy is reduced. Figure 2.7 illustrates the basic concept. The architectural design describes a set of layers being responsible for one specific behaviour each.

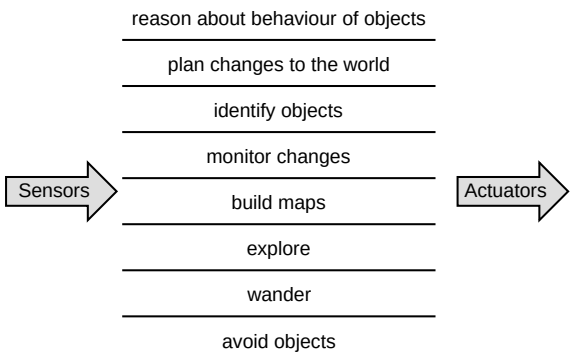


Figure 2.7: *Brook's* subsumption architecture [60]

Application of the Concept

Since SPA served as a reference model for years, several different systems based on the concept can be found in literature [61]. One prominent example has been presented by *Shanahan and Witkowski* and uses a *Khepera* robot [62]. The authors describe two applications in detail that are used for navigation and map building within an office scenario. The navigation part results in knowledge about open and closed doors, while the map building describes the rooms' and doorways' layout. A similar example is given by *Stentz* [63]. Both examples are purely based on the SPA concept.

Additionally, two famous approaches have been presented applying *Brook's* basic design to real robots: *Genghis* [64] and *Herbert* [65]. An overview of the fundamental concepts is given in [66]. The first example, *Genghis*, is a six-legged walking-machine with decentralised control. The overall movement of the system is based on the reactions of the legs. Each leg is controlled by two motors (one for up-down and one for back-forth movements). The system consists of augmented finite state machines (FSM) for each behaviour. The behaviours are built incrementally and can be run by selectively deactivating later FSMs. Furthermore, *Connell* presented the autonomous and mobile robot *Herbert*, which is able to wander around in a closed office environment [65]. In this context, *Herbert's* task is to get into peoples' offices and pick empty soda cans from their desks. *Herbert* is able to avoid obstacles, to follow walls, and to recognise objects looking like soda cans or desks in real-time. In total, the behavioural repertoire consists of 15 different abilities that are ordered using *Brook's* layered approach.

Characterisation of the Approaches

Although the discussed architectural approaches and design patterns have differences, they are built upon each other and can be considered as one major solution framework. Thus, the list of requirements can be considered as follows:

A) Adaptivity: The approach allows for a restricted **adaptivity** – it takes the environmental conditions into account.

B) Robustness: The approach is **robust against disturbances** that have been foreseen at design time and taken into account when designing the finite state machines; unanticipated situations might lead to failures.

C) No interference with the system's logic: The design of the system relies on defining explicit behaviours realised as finite state machines. Hence, integrating existing systems **without interfering with the system's logic** is not feasible.

D) Operability: Due to the strong interconnection between control mechanism and element under control, the underlying process might not **remain operable** in cases where the control mechanism fails.

E) Flexibility: The concept does not cover **flexible goals**.

F) Vast situation and configuration spaces: Perceiving real-world environments using sensors leads to **vast situation spaces**. In contrast, the configuration space in traditional robotics is very restricted. As a result, **unanticipated situations** lead to failures in most cases.

G) Self-improvement: Current research addresses questions of **self-improvement** within the domain of robotics (see e.g. [67]), but the topic is neglected within the design pattern.

H) Restricted exploration: Since all behaviours are predefined, the approach does not make use of automated **exploration mechanisms** to identify better behaviours – consequently, no controlled and restricted usage of such mechanisms is considered.

I) Decentralised operation and collaboration: Decentralised collaboration is not explicitly covered by the design.

J) Comprehensibility: The **comprehensibility to users** depends highly on the purpose of the system, the particular model (SPA), or the number of possible behaviours – in complex systems, comprehensibility can be considered as problematic.

K) Real-world requirements: Since robots are designed to take part in daily life and to fulfil specific tasks in this context, the corresponding architectures have to cope with **real-world requirements**.

L) Generalised approach: State machines are designed to cover just one task. Hence, the basic approach is not a generalised concept – it requires high effort to customise a certain solution.

In general, SPA has a further disadvantage: the model-based concept is very slow. In particular, the system can almost never plan at the same rate as the environmental conditions are changing. In addition, all modules depend on each other and failures affect the whole system. Furthermore, the open loop plan execution has been identified as inadequate due to uncertainty and unpredictability [68] – changing environments require an adaptation of the system's world model. The behavioural approach does not take such a world model into account – consequently, the realisation of behaviour is easier to develop than standard SPA. But in this concept, planning and optimisation are more difficult to be taken into account.

2.3.3 Multi-Agent Systems and Adaptive Agents

A multi-agent system (MAS) is a technical system composed of multiple interacting, intelligent entities – the so-called *agents* [69]. Typically, the term MAS is used for software agents, but it can also refer to robots, humans, or teams containing a mixture of both. These agents are used to model or solve problems that cannot be handled in a standard monolithic way due to high complexity. Usually, a MAS builds some kind of heuristic approach for an insolvable or very complex problem [70]. In literature, the concept has been successfully

applied to several well-known tasks, e.g. modelling social structures [71], disaster response [72], or on-line trading [73].

Based on the terms given by *Wooldridge* [69], agents in a MAS can be characterised using three main attributes: a) *autonomy* – each contained agent is at least partially autonomous, b) *local view* – none of the agents has a global view of the system (e.g. due to the system's complexity), and c) *decentralisation* – all agents are equal, especially no designated controlling agent exists (otherwise the system can effectively be reduced to a monolithic system [74]). Although these characteristics define restricted behaviours based on simple rules, a collection of collaborating agents (the *multi-agent system*) can manifest self-organisation and complex behaviours.

Since research has investigated MAS for several years, a wide variety of architectures and approaches to design adaptive and self-organised behaviour has been proposed [69, 6]. First attempts of *deductive reasoning agents* were based on logics [75, 76], but their popularity decreased quickly due to several restrictions and limitations of the concept [77]. Afterwards, the so-called *mental* aspects of the agents have moved into the focus [78, 79, 80]. As a result, decision making does not completely found on pure logics. Among these approaches, the *Belief-Desire-Intention* (BDI) model [78] has been established and widely accepted as basis for further research in this domain.

The Belief-Desire-Intention Model

BDI implements the principal aspects of *Bratman's human practical reasoning* theory [81] and is characterised by the eponymous three attributes: the agent's *beliefs*, *desires* and *intentions*. An agent uses these basic concepts to solve a particular problem. The process of developing or choosing a plan or strategy is separated from the execution of the currently active plan or strategy. Due to this differentiation, the agent is able to react on observed changes in the environment immediately – usually, deliberating plans takes time (choosing what to do is often referred to as *optimisation problem*).

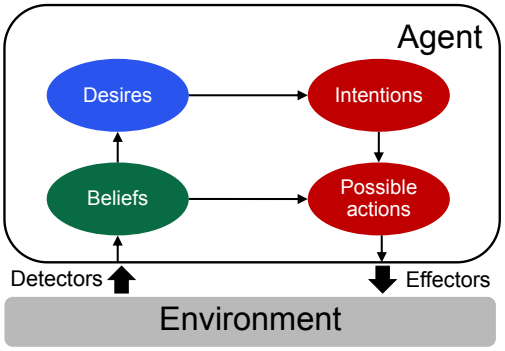


Figure 2.8: The BDI design model [78]

Based on the general model of the architecture (see Figure 2.8), the designer of an agent has to define the particular mental attitudes. The *beliefs* are used to represent the current status of the system as the agent observed it – including the environment and other agents in the neighbourhood. In addition, beliefs can also contain inference rules – this attribute allows for forward-chaining and leads to new beliefs. The term *belief* is used on purpose instead of knowledge, because the belief of the agent is based on its

perception, which is not necessarily correct. The local view in combination with incomplete or possibly disturbed sensor information leads to uncertainties in modelling the world's status.

The *desires* represent the agent's motivation – in particular, each agent has personal goals which are tried to be fulfilled at runtime. Typically, agents are not designed to cover just one goal – the desire is a trade-off between several different (and possibly conflicting or inconsistent) goals. Thirdly, the architecture defines an *intentions* entity. Intentions are the result of desires and the current observation – the agent has chosen to achieve a specific desire. Within the implementation of the BDI model, intentions are defined as effectively started actions from a selected plan. In this context, *plans* are actions or sequences of actions agents can perform to achieve their intentions. *Bratman* defined plans as only being partially conceived, with details being filled in as they progress [81]. In order to trigger the agent's reactive activities, *events* are used. Such an event can be applied externally via sensors or embedded systems, or internally to activate updates or activity plans. Usually, they are used to update beliefs, trigger plans, or modify goals.

Application of the BDI-concept

Several representatives for architectures using BDI concepts are known in literature. As one popular example, *Martin et al.* introduced the *Open Agent Architecture* (OAA) [82], which aims at enabling the allocation of software services through a distributed set of autonomous agents. Communication between and cooperative effort of the agents are organised by one or more brokers. These brokers have knowledge about the capabilities of other agents and are responsible for the matching of requests from users and agents. The goal of the concept is to enable a transparent view on the system. For a particular request, a user or agent does not have to know the identities, locations, or number of involved entities. The OAA is used as a framework to minimise the effort needed for achieving interoperability between various platforms, or the consequences of dynamic environments. Although it aims at being a generalised approach to agent architectures, the main concerns are software services and collective task performing.

The *Adaptive Agent Architecture* (AAA) approach presented by *Kumar et al.* [83] has similarities to the previous example. The main difference is the focus, which has been set on *teamwork*. Based on theoretic analysis, the authors developed a brokered architecture, which allows for the automatic recovery from failures (of the broker). This broker is responsible for forming groups based on the tasks and requests he receives. The self-organisation of the agents aims at providing *self-healing* capabilities. In contrast to the system targeted by this thesis, both systems lack properties like self-optimising behaviour, learning capabilities, architectural design, and a re-usability of other systems.

The Cognitive Agent Architecture (COUGAAR) is an open source, distributed agent architecture [84, 85, 86]. Due to the financial support of the Defense Advanced Research

Projects Agency (DARPA), the focus has been set on developing distributed systems with a large number of involved agents. Hence, these systems are characterised by military needs like hierarchical task decomposition. Additionally, recent extensions of the approach investigated fault tolerance, scalability, and security aspects. Based on the vision of MAS, the agents are supposed to achieve a common goal by arranging themselves into a society, which collectively solves the tasks. The COUGAAR framework contains an adaptive control mechanism using an *Adaptivity Engine* that makes use of the agent's cognition and aggregates the observed attributes by the integrated *metrics service*. Each agent has different operational modes that can be exchanged depending on the situation.

Characterisation of the BDI Approach

Considering the list of requirements as introduced before, the MAS frameworks and the corresponding BDI concept can be characterised as follows:

A) Adaptivity: For the agent itself, the BDI concept **enables adaptivity** according to some predefined characteristics.

B) Robustness: According to the specific agent model, the system is characterised by a restricted degree of **robustness against disturbances** – to cover disturbances that occur according to unforeseen situations, concepts like self-improvement are missing.

C) No interference with the system's logic: The concept relies on building integrated agent-oriented systems – thus, it is not able to control existing solutions **without interfering with the system's logic**.

D) Operability: Since the agent covers both, the logic and the adaptation, the basic purpose of the system will not **remain operable** in case of failures of the control mechanism.

E) Flexibility: As the *desires* are predefined for each autonomous agent, the approach does not cover **flexible goals**.

F) Vast situation and configuration spaces: In most agent systems, **vast situation and configuration spaces** are not considered; the same observation can be made for **unanticipated situations**. But – in principle – (especially if an appropriate model exists) both can be handled in the context of the design.

G) Self-improvement: Although some work has been done on learning in MAS (cf. [87, 74]), the basic concept does not cover **self-improvement**.

H) Restricted exploration: Due to using predefined desires and behaviours, agents have no **exploration mechanism** to identify new behaviours – consequently, such a mechanism is not subject to restrictions in its usage.

I) Decentralised operation and collaboration: Agent systems are typically designed to enable **decentralised collaboration**, especially variants like AAA [83] have their main focus on this topic.

J) Comprehensibility: The **comprehensibility to users** depends on the complexity of the system. Since the motivation of this thesis is founded on the observation that systems

become too complex for standard design techniques, BDI is assumed to be not applicable to the corresponding systems.

K) Real-world requirements: Typically, agent systems are simulation-based abstractions of real problems [70] – thus, they can cope with *abstracted* **real-world requirements**.

L) Generalised approach: The BDI design pattern formulates a generalised approach for a certain category of systems. Additionally, it integrates large parts of the system's functionality and is therefore not generally applicable.

Due to the abstract focus, the concepts developed for MAS are adapted in other research initiatives like AC or OC in order to solve real-world problems, rather than just artificial or scientific questions [88]. The BDI approach relies on the existence of goals, but an explicit representation of these goals is missing in most cases [89]. In contrast to the system presented in this thesis, BDI lacks mechanisms to learn from past actions, decisions, and observations or to react on unforeseen situations. In particular, the architecture does not possess an explicit component to cover self-optimising behaviour [90, 91]. Generally, it is difficult to couple learning and the BDI architecture, since the model does not cover interaction principles among agents of the population [92]. Besides learning questions, the approach has weaknesses when being applied to real-world problems: the decisions are inferred using multi-modal logics, which can hardly be applied to this problem domain due to incomplete axiomatisations and inefficient computability [6, 93]. Furthermore, these inference rules have to be developed in advance – here, a more autonomous solution would be useful. In turn, scientists have not agreed on the necessity of the three BDI attitudes. In traditional decision theory the distribution of responsibilities is questioned, while in artificial intelligence researchers argue that three attitudes are insufficient [90].

2.3.4 Autonomic Computing

Based on the ideas of *Autonomy Oriented Computing* (AOC) [94], IBM developed the "Vision of Autonomic Computing (AC)" [1]. AOC focused on the development of artificial technical systems capable of imitating behaviour observed in nature (e.g. ants) in order to cope with computational-intensive problems. *Jin and Liu* defined four key aspects for a new type of autonomy-oriented systems: autonomous entities, emergent behaviour, adaptive solutions, and self-organised behaviour. The basic ideas of AOC served as input for the vision of AC. The *AC Initiative* by IBM adapted the concept and applied it to the domain of computer infrastructures and large server farms.

Like OC, the AC initiative is motivated by the observation of rapidly growing complexity of computing systems and the insight that their management needs new visions for designing systems. In particular, systems capable of self-management are desired. Driven by the economical motivation to reduce cost and effort, the reduction of complexity is realised by increasing the freedom for self-managed entities. In this context, self-management is achieved

by four basic attributes [3]: AC systems have to be **self-configuring** (be able to automatically configure their components), **self-healing** (faults and misbehaviour is automatically detected and corrected), **self-optimising** (capable of monitoring the own performance and adapting to the measured values), and **self-protecting** (able to pro-actively identify and protect against arbitrary attacks). Following *Sterrit's* approach [3], these four attributes are achieved by self- and environmental-awareness in combination with a control loop defined by self-monitoring and self-adjusting components of the system. The common architectural pattern for AC systems enabling these attributes is the so-called *MAPE cycle*, which is presented in the following part.

The MAPE Cycle

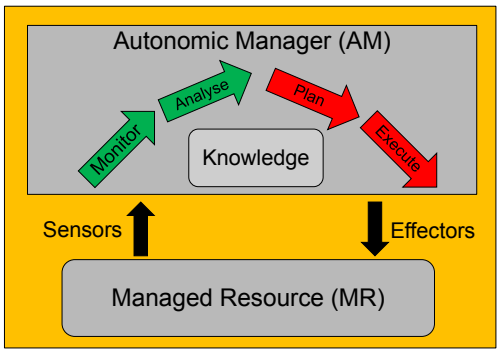


Figure 2.9: The MAPE cycle [1]

With the MAPE cycle, IBM introduced a basic architectural concept for autonomic systems [1, 95]. Within this abstract information framework for self-managing IT systems, an autonomic system is defined as a set of autonomic elements (AE) with each AE consisting of an autonomic manager (AM) and the managed resource (MR). The MR has to provide a management interface (defining the two attributes *sensors* and *effectors*), which is used to perform the communication between the AM and its MR. In order to allow for an adaptation of the MR to changing environmental conditions and reacting on failures, it has to be observable via these sensors. Additionally, it has to be adaptable using the effectors to allow for changing the MR's behaviour. The AM itself is designed as a control loop, providing capabilities to perform the four tasks ***m**onitor*, ***a**nalyse*, ***p**lan*, and ***e**xecute* (the *MAPE cycle*, see Figure 2.9). Additionally, the MAPE cycle takes supporting knowledge of the environment and management policies into consideration. Compared to the OC approach as presented in Section 2.2, the MR represents the *System under Observation and Control*, while the MAPE cycle covers the tasks of the Observer/Controller component. In addition, AC systems following the MAPE concept are *closed control loops* (cf. Figure 2.5(b)) in terms of control theory (see Section 2.3.1). The self-managing system observes the status of some resources (software or hardware components) and autonomously controls their parameters in order to keep them within a predefined range.

Similar to OC, the desired behaviour of the AM component relies on high-level business goals given by the system's user in form of policies. Examples for such goals include maximising a given utility function or keeping the system's performance within certain boundaries [1]. In addition, these abstract goals can be split into more specific simpler

ones by forming a hierarchy of control loops: some autonomic managers can manage others and these can directly manage resources. As a result, a management hierarchy similar to business organisations can be built. Summarising, the approach proposes a design of interactive collections of autonomic elements. Individual entities are responsible for managing resources (e.g. devices or groups of other autonomic elements) in order to cover parts of the management process automatically without the need of a user. Instead, the systems self-configure and adapt themselves in accordance with given policies from humans or other higher-level elements.

Application and Extensions of the Concept

Since 2003, a variety of architectural frameworks based on the MAPE cycle and consisting of self-organised autonomic components has been proposed. Besides theoretical considerations (e.g. *White et al.* defined general requirements for AC architectures [96] and *Koehler et al.* searched for a computational model that allows for guaranteeing behaviours [97]), the most important representatives of AC focused on investigating the design and the process of the autonomous control cycle. As one example, *Fuad and Oudshoorn* presented a system architecture for such an autonomic element [98]. Based on the initial MAPE cycle, they describe the structural operation, since this is the fundamental building block of any autonomic system.

The basic MAPE cycle is widely accepted in the AC community at design level. It serves as basis for several installations and prototypes in research ranging from a uniform framework for automated management of Internet services and their underlying network resources (the Autonomic Service Architecture [99]) over a J2EE-based software architecture for adaptive webserver applications [100] to systems for autonomic grid applications (*AutoMate* [101]). Furthermore, *Liu and Parashar* presented *Accord*, a programming framework for autonomic applications [102]. A prototype installation in the context of large server farms has been presented with *Unity* [103]. The autonomous behaviour of AC systems is based on high level goals in terms of policies. *Calinesu* worked on the definition and description of these policies [104] and presented a model-driven autonomic architecture [105, 106]. The framework for the formal specification of autonomic computing policies has the disadvantage that the developed approach is very specific and consequently hardly transferable to other domains.

Characterisation of the MAPE Concept

Considering the list of requirements as introduced before, the *MAPE concept* and its current applications can be characterised as follows:

A) Adaptivity: Obviously, the whole approach is based on the goal to **enable adaptivity**.

B) Robustness: Since the particular solutions of the general concept are based on predefined behaviours, the **robustness against disturbances** is restricted. In particular, it depends on which policies are foreseen by the developer as a discovery component is missing.

C) No interference with the system's logic: In general, the MAPE control loop is able to control existing systems **without interfering with their logic**, but an investigation of how this application can be conducted is missing. Thus, the idea has been formulated but not yet sufficiently realised.

D) Operability: The MAPE cycle postulates an additional control loop – thus, it should be possible that the managed device **remains operable** in cases where the control mechanism fails.

E) Flexibility: Whenever goals are discussed in the context of AC, they are referred to as static policies (e.g. [1]). Hence, **flexible goals** are not considered, but – in general – they are consistent with the concept.

F) Vast situation and configuration spaces: AC's predefined strategies are hardly applicable to **vast situation and configuration spaces**. In addition, completely **unanticipated situations** cannot be covered.

G) Self-improvement: The need for **self-improvement** and the corresponding autonomous learning capabilities within the architectural concept is not emphasised.

H) Restricted exploration: MAPE's degree of freedom does not include **exploration** phases – consequently, there is no need of restricting the trial parts due to the usage of predefined logic and policies.

I) Decentralised operation and collaboration: Initially, *Kephard and Chess* wrote that an autonomous element "may require assistance from other elements to achieve its goals" [1], but a systematic investigation of patterns for **decentralised collaborative behaviour** is missing to the author's knowledge.

J) Comprehensibility: The **comprehensibility to users** depends on the complexity of the autonomic system. Especially in large systems, it might suffer if the required policies have to take complex interdependencies into account.

K) Real-world requirements: AC systems are mainly applied to infrastructure – thus, they fulfil **real-world requirements**.

L) Generalised approach: At an abstract level, the MAPE design is generally applicable. Due to the strong focus on IT infrastructure and mainframes, proof for a transferability to other domains is missing and concepts for other technical systems have not been developed, yet.

Comparing the focused goal of this thesis' system (and its context OC) with the one of AC shows the main difference between both approaches: the application field. AC deals with an increased heterogeneity, interconnectedness and complexity of IT-systems (namely large-scale enterprise server systems [8]), while OC covers a variety of technical systems. Main purpose of AC has been to build market-ready solutions used to decrease the effort for

managing IT-infrastructure, rather than developing a generalised approach to controlled self-organisation in technical systems. In addition, learning and exploration of new behaviours automatically has not been explicitly investigated by the AC community.

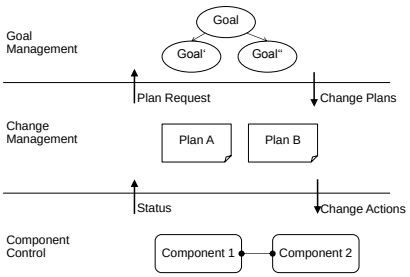
2.3.5 Further Architectural Approaches

The above presented examples cover the major part of research focused on designing adaptive systems. Of course, these are not the only domains dealing with aspects relevant for this thesis. The following part gives a short insight into other related research domains and loosely connected concepts. Considering the motivation to enable context-awareness, adaptivity, and self-organisation, further research areas have strong connections to these topics. As one example, the *Autonomic Communication* [107] initiative has emerged recently covering similar goals compared to IBM's Autonomic Computing proposal. The most significant differences are that it focuses on individual network elements, studies how the desired element's behaviour is learned, influenced or changed, and how it affects other elements within this network. Thus, it has a strong connection to the Organic Network Control system as presented in Chapter 7.

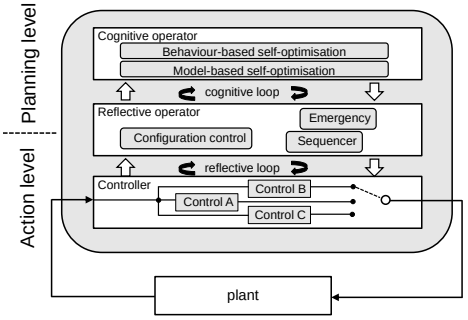
Furthermore, *proactive computing* [10] has to be named that served as foundation and pool of ideas for current initiatives like OC or AC. Considering the aspect of highly interconnected devices surrounding everybody, *ubiquitous computing* [108] and *pervasive computing* [109] share parts of the same motivation and vision. But since they follow a different target definition, the architectural context of this thesis is not matched.

In the context of distinguishing between appropriate on-line reactions based on a fixed set of policies and a separated planning component, further contributions from software engineering, mechanical engineering, and agent systems have to be mentioned: a three-layered architecture for self-management [110], the *Operator-Controller Module* [111], and the *Anytime learning* concept [112]. The former one – the *Three Layer Architecture Model for Self-Management* (3LA) – has been presented by *Kramer and Magee* in [110]. Figure 2.10(a) illustrates the basic design. The system consists of a given goal and a set of software components providing the required logic. The target of the self-management process is that these components configure themselves according to given goals – if this is not possible, they have to be capable of reporting the problem. On the lowest layer (the *component control*), the system's components are interconnected and provide the static operational functionality. In case of malfunctions or changes in the bottom layer's status, the middle layer is responsible for executing a predefined sequence of actions in order to adjust the bottom layer's system to the new conditions. Due to these predefinitions, a fast reaction is guaranteed. In cases where no adequate plan exists or the goals are changed, the highest layer is responsible for planning new strategies. Current research focuses on deriving local strategies for given conditions from more abstract global goals. The main problems of this concept are described as

finding a formal description for each situation, a powerful deriving engine, and a possibility to derive specific goals from abstract directives. In addition, feedback on how the on-line system performed is not used to self-improve the behaviour.



(a) Three layered architecture model for self-managed systems [110]



(b) The Operator-Controller Module [111]

Figure 2.10: Multi-levelled learning

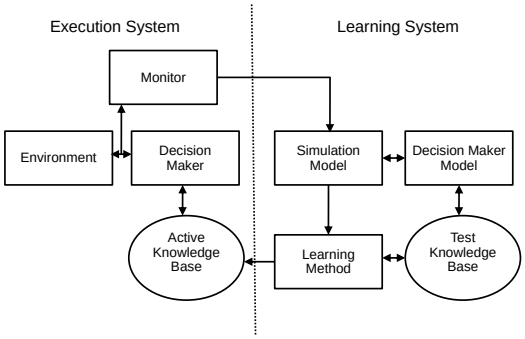


Figure 2.11: Anytime Learning [112]

In mechanical engineering, the *Operator-Controller Module* has been investigated as one approach to realise self-optimising systems [113]. Motivated by the increasing number of errors caused by the growing functional repertoire of mechanic systems [111], a highly integrated self-improving system has been developed. On the lowest level, the *operational controller* is responsible for processing the productive system in hardware. Based on a closed control loop concept, the middle layer contains a *reflective operator* in software that can adapt the operational controller based on a given set of policies. Finally, the highest level contains a *cognitive operator* monitoring the level beneath. It is responsible for gathering information on itself and its environment. Therefore, various methods such as learning, use of knowledge-based systems, or model-based optimisation in order to self-improve the behaviour are used. Due to the fully integrated control-loop-based system design, failures of the learning module affect the whole system. In addition, the approach is highly domain-specific and considers learning only as determining new policies. Thus, it is assumed that each of these policies is optimal (which makes the simulation model absolutely reliable) – consequently, self-improvement by taking feedback on the applied policies into account is not considered.

Finally, the *Anytime learning* concept by *Greffenstette and Ramsey* [112] can be considered as inspiration for two-layered learning. In the context of investigating machine learning

techniques for sequential decision problems, the authors developed their *SAMUEL* learning system [114, 115]. *SAMUEL* is designed to learn adequate reactive policies, described as condition/action pairs, based on a given simulation model of the environment [112]. The purpose of this simulated environment is to evolve new rules that are optimised using an Evolutionary Algorithm [116]. This coupled learning module is integrated into the productive system as illustrated in Figure 2.11. The learning module is continuously executed. It controls the interaction of the productive system with the environment, adapts its simulation model, and explores new policies. Such new policies are provided to the execution module. The process assumes that the model-based simulation will always lead to appropriate solutions and does not cover an on-line improvement of the action-selection policies. A similar approach has been proposed by *Oreizy et al.* in [117]. Additionally, hierarchically organised LCS variants like *ALECSYS* [118] and *MonaLysa* [119] are known – in contrast to the framework investigated in this thesis, they solely focus on increasing the learning speed, rather than incorporating safety restrictions.

2.4 Summary

Based on the initial requirement analysis for the targeted system, this chapter discussed the applicability of existing design patterns, their corresponding frameworks, and their prominent representatives to these requirements. In the first step, OC's *generic Observer/Controller design pattern* has been identified as a basic platform for the developed framework. The concept's main control loop and its major components have been explained as well as possible distribution variants and current applications. Afterwards, related architectural approaches have been discussed and compared to the initially formulated requirements.

The analysis of the state of the art shows that none of the discussed concepts is applicable to close the identified gap. In particular, no alternative approach fulfils all the requirements sufficiently. Especially, a solution for restricted on-line learning in safety-critical environments is missing. Although different approaches to distinguish between learning tasks exist, no explicit investigation of automatically improving the system's behaviour while simultaneously guaranteeing an appropriate system performance has been identified. It is the goal of this thesis to present a solution that is capable of both: fulfilling the requirements and presenting an exemplary on-line learning concept for safety-critical systems. The following Table 2.1 summarises the discussed results. The table distinguishes between five different classifications: "XX", "X", "0", "$-$", and "$--$". The first one denotes a full match of the requirement by the particular technique, "0" states that the requirement might be satisfiable (but – to the author's knowledge – has not been appropriately investigated, yet), and "$--$" states that the requirement is not fulfilled. The remaining two classifications are nuances within this scale. Within the next chapter the developed framework is presented, which is characterised by fulfilling the initially defined requirements.

ID	*Criterion*	*OC*	*CT*	*Robotics*	*MAS*	*AC*
A)	**Adaptivity**	*XX*	*X*	*X*	*X*	*XX*
B)	**Robustness**	*XX*	*X*	*X*	*X*	*X*
C)	**No logic-interference**	*X*	−−	−−	−−	*X*
D)	**Operability**	*XX*	−−	−	−−	*XX*
E)	**Flexibility**	0	−−	−−	−−	0
F)	**Vast spaces**	*XX*	0	0	0	0
G)	**Self-improvement**	*X*	−	0	0	0
H)	**Restricted exploration**	*X*	−−	−−	−−	−−
I)	**Decentralised collaboration**	−	−−	−−	*XX*	0
J)	**Comprehensibility**	*X*	−	0	−	0
K)	**Real-world requirements**	*XX*	*XX*	*XX*	0	*XX*
L)	**Generalised approach**	0	*X*	−	0	0

Table 2.1: Classification of the state of the art

Chapter 3

System Design

This chapter presents the developed framework and the corresponding design of the system. Thus, the main focus is set on the functional concept and the contained main components, their co-operation, and their responsibilities. The investigation, which particular techniques are used to fill out the needs described by these components, is subject of Chapter 4. The chapter is divided into four basic parts. Initially, the system's objective is defined. The desired behaviour of the system is designed using an adapted variant of a classification from the OC domain (see *Schmeck and Müller-Schloer* [120]). Based on this definition of what the system has to be able to do and how it is supposed to behave, the scope of the system is discussed. Here, the question is for which existing technical solutions the architecture and the resulting framework are applicable. Therefore, requirements for systems to be controlled by the framework are introduced.

Furthermore, the architecture itself is presented. Based on an abstract overview of the design, the particular layers and their functionalities are presented. Thereby, a special focus is set on the necessary adjustments needed to enable the observation and control of a new system - in particular, the question "What has to be done in order to wrap a system with the developed framework?" is brought up. This is of special interest, since the solution presented within this thesis aims at providing a generalised approach for varying types of systems. Consequently, the tasks for the adjustment process have to come with low effort and considerable knowledge about the underlying problem.

3.1 Target Definition

Based on the terminology given by *Schmeck and Müller-Schloer* in [120], the target of the framework is defined in the following section, which is based on previous work presented in [121]. Let S be a parametrisable productive system controlled by the framework (the control mechanism CM). Figure 3.1 illustrates the context. S is the productive system and the combination of S and its CM is denoted as S'. The combined system S' operates in an *environment*, which is subject to changing conditions.

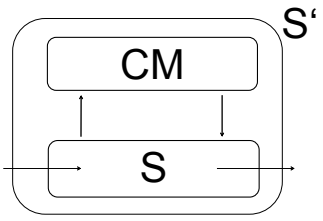


Figure 3.1: Controlled system and control mechanism

Definition 1 (Environment) *The environment aggregates all entities outside of S and its control mechanism CM. It consists of four types of attributes: local, receivable, global, and inaccessible.* ***Local*** *attributes are completely accessible by CM, while* ***global*** *attributes need some higher instance with a broader focus (e.g. a hierarchical element).* ***Receivable*** *attributes are observed locally by a neighbouring system N and communicated to the CM of S. In contrast,* ***inaccessible*** *attributes are hidden and neither observable by CM, nor by higher entities.*

The system S is continuously performing its productive tasks considering its configuration defined by the set of parameters. Typically, these parameter configurations are static – the CM's goal is to adapt these parameters dynamically. One parameter set describes one possible configuration of the SuOC. Consequently, the *configuration space* can be defined as follows:

Definition 2 (Configuration space) *All possible parameter configurations of the SuOC define the* ***configuration space*** *CS (also called* ***parameter space****). Two types of parameters are distinguished: accessible and non-accessible (with respect to an external CM). In the remainder, all relevant parameters are assumed to be accessible. The current configuration of the SuOC at time t can be expressed as $c(t) \in CS$ with c being one possible instance of C.*

The dynamic adaptation of S has to be performed according to changes of the environmental conditions: the *situation*. CM is responsible for monitoring all attributes of S and all attributes of the environment affecting the performance of S, which results in a description of the current situation.

Definition 3 (Situation and state space) *The SuOC's state can be observed using internal and external variables defining the* state space *Z. All environmental attributes with impact on the SuOC's performance define the situation space Y, with the current status of*

*these environmental attributes being expressed as $y(t)$ (also denoted as the SuOC's **situation**). The current state $z(t)$ of the SuOC consists of the status of the internal parameters $c(t)$ and the status of the external environmental attributes $y(t)$. This can be expressed as $Z = CS \times Y$.*

According to Definition 3, the configuration of S at time t can be defined as a function $z(t)$ – for instance, if n attributes are used to describe the state of S, $z(t)$ is a vector in some n-dimensional state space Z. Some of these parameters are used as evaluation criteria (also called *objectives*) $\eta_1 \dots \eta_k$, which are provided by a higher-level external entity (the user). The CM depends on a user-specified goal g, which is expressed by the objective function f. The system can incorporate a set of different goals G such that the user can choose a goal $g \in G$. At runtime, the CM searches for the best mapping between an observed situation $y(t)$ (or $z(t)$) and a parameter configuration $c(t)$: $y(t) \rightarrow c(t)$. Thus, it has to maximise performance of S according to the current goal: $f_g(z(t)) \rightarrow max$.[1] The function f uses the evaluation criteria $\eta_1 \dots \eta_k$ and maps the aggregated system state $z(t)$ into the set of real numbers by estimating the current system performance. Typically, the functional space of f is known in advance, including its maximum and minimum boundaries (e.g. if S is a Peer-to-Peer protocol instance and the evaluation function refers to maximising the download rate, the minimum of the functional space is equal to zero and the maximum is given by physical characteristics of the channel). Based on the objective function f, a hierarchy of subspaces of the space Z can be built characterising the system's performance:

1. Target Space (TS): If the goal g of the CM is fulfilled, e.g. the performance quantified by $f(z(t))$ is above a given threshold θ_t, the corresponding state $z(t)$ is part of TS. Thus, TS is the set of states where no control actions of CM are needed.

2. Acceptance Space (AS): The system state $z(t)$ is called *acceptable*, if an acceptance criterion or threshold θ_a is satisfied: $\theta_a < \theta_t$ if the fitness function is to be maximised, $\theta_a > \theta_t$ otherwise. The set of all acceptable states satisfying the threshold θ_a is called *acceptance space* (AS). Obviously, TS is a subset of AS. Typically, the standard static configuration of S (without additional CM) will lead to acceptable states on average. Although the system's state is acceptable, the CM will try to find a better solution and consequently reach TS.

3. Survival Space (SS): If S is in an unacceptable state, but it is still possible to modify the system state $z(t)$ such that at some later time t' the resulting state $z(t')$ is acceptable, the system state belongs to SS. For example, let S be an instance of a data communication protocol – CM could have changed its queue sizes to zero and no packages are stored anymore (and consequently none are processed). As a consequence, the system performance will be low – but it can return to AS by increasing this value.

[1] This is equivalent to a goal specifying a minimisation of f, which can be solved by maximising $-f(z(t))$

4. Dead Space (DS): If S cannot return into an at least acceptable state *by itself*, it is not part of one of the previous sets and therefore belongs to the Dead Space. For instance, a system running out of energy (battery-mode) is in DS until an external authority tackles the problem.

Based on this enumeration, the target of the CM is to adjust $c(t)$ depending on the observed state $z(t)$ in such a way, that at the next evaluation time t' the corresponding system state $z(t')$ will be part of TS. The remainder of this chapter explains, how this is achieved in detail by explaining the architecture and its components.

3.2 Scope of the System

The following part of this chapter defines which systems can be controlled by the CM. Since a real-world applicability is of major focus, the typical key-characteristics of real-world technical systems are discussed initially, based on the definition given by *Rao and Georgeff* [6]. Afterwards, the effect of these characteristics on the CM's scope is explained. The aspects given by *Rao and Georgeff* are used as a rough guide for the purpose of the proposed framework:

- *Environment:* The environment is non-deterministic. It changes at each instant of time and can evolve into potentially many different ways.
- *System:* Besides the environment, the technical system (the combination of S and CM) itself is non-deterministic. At any instant of time, the system can be instructed to accomplish potentially many different objectives.
- *Actions:* The CM's best action depends on both, external stimuli (context-aware) and the internal status of system S.
- *Horizon:* The system's horizon is restricted to locally available sensor data, no global information can be used to decide on which is the best action to be taken.
- *Cycle:* The processing of CM and its actions are performed in cycles that correspond to the rate at which the environment evolves.

These real-world aspects affect both: the system S and its control mechanism CM provided by the proposed framework. Thus, only those systems are of interest that are situated and processed in dynamic environments. This aspect corresponds to the requirement of performing an adaptation according to changes of the environmental conditions and not only according to internal changes. Furthermore, a possibility to access this environment through sensors has to exist in order to allow for a suitable monitoring of the situation. This monitoring is the basis for the decision about adapting S to observed changes – the *context-awareness*. Hence, the effects of changes in the environmental conditions have to

appear in such a form that the CM can adequately react on them – which means that effects and actions of CM have to be within similar time-boundaries.

The CM has to work on locally available information only. Based on this *locality* aspect, some restrictions regarding the applicability of the framework to technical systems can be derived. On the one hand, the performance of a technical system S to be controlled by CM has to be measured locally without additional global knowledge. On the other hand, S has to be configurable on-line (at runtime) using a set of parameters from the configuration space CS. The configuration of the variable parameters under control of the CM needs to have impact on the performance of S – otherwise, such a situation-dependent adaptation would be of no avail. In addition, this impact of the parameters on the performance has to be deterministic (at least to a certain degree). The approach relies on drawing conclusions about the past actions and the corresponding change in the objective function's payoff. If the impact is more random than deterministic, deriving such kind of knowledge is not possible. Furthermore, the objective function f itself has to be deterministic, otherwise deriving knowledge is again not possible. In addition, f has to work on locally observed or received attributes of the environment and of S (the state $z(t)$). Other global or inaccessible attributes cannot be taken into account in a distributed system (cf. Definition 1).

In addition, the quality analysis using the objective function f is continuously performed by the CM. Thus, f has to possess characteristics like being *continuously differentiable*, having a *simple computability*, and *low structural change over time*. The requirement that the function has to be continuously differentiable is formulated due to the safety-based learning concept, which is realised by considering actions of "nearby" situations as appropriate. Details on this assumption are given in the remainder of this chapter. The function f will be frequently computed with the necessity of an immediate reaction – which restricts the set of possible functions for f to those with *relatively simple computability*. In order to enable learning within the CM, an automated acquisition and storage of knowledge is needed. The learning relies on finding appropriate mappings between situations and the corresponding actions. This implies that appropriate actions for a given situation exist and these actions will also be correct, if the corresponding situation occurs again. A time-dependence of the basic fitness landscape modelled by f can be covered if it is quasi static, i.e. it does not completely change the landscape (only slightly changing landscapes are manageable since the learning takes time).

The aspects discussed before serve as indicators to decide whether an existing technical system S can be controlled by the proposed framework or not. If S fulfils all requirements and is characterised by the discussed attributes, it can be equipped with the additional CM and therefore adapted on-line to changing environmental conditions.

3.3 System Architecture

Based on the goal and the scope of the system, an architectural design of the framework has been developed that serves as basis for answering the research questions as brought up at the beginning of this thesis. In the remainder of this chapter, the general architecture defined by the framework is discussed. In the further course of this thesis, the framework is applied to different application scenarios, which requires a customisation and corresponding application-specific modifications of both – the framework and its architecture.

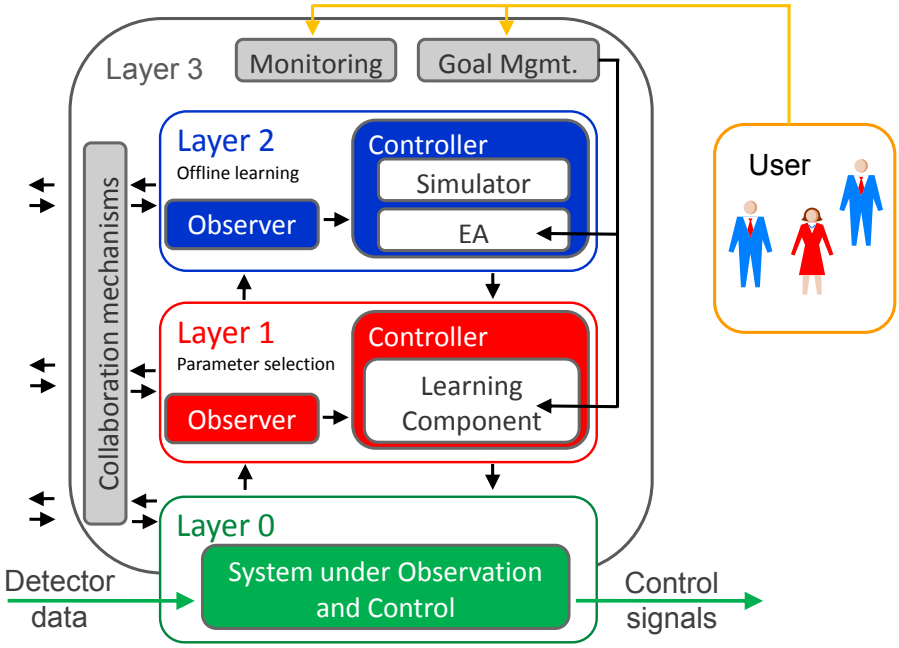


Figure 3.2: System architecture (single node)

Figure 3.2 depicts the general architecture of the proposed framework which is based on the *generic Observer/Controller design pattern* as introduced in Chapter 2.2.1. The design distinguishes between four consecutive layers. The bottom layer (Layer 0) encapsulates an existing technical system S, e.g. the before mentioned traffic control system. In terms of OC, S is called *System under Observation and Control* (SuOC) [25]. The particular internals of the system's logic and insights on the corresponding domain are not needed. However, it is required that the parameters of the encapsulated SuOC can be altered by the Layer 1 component and the status of the system and its environment are accessible to higher layers.

In combination with the component situated at Layer 1 of the architecture, Layer 0 forms a *control loop*. The Observer of Layer 1 collects local information and current settings of the SuOC and aggregates them into a vector describing the current *situation*. This situation vector then serves as input to the learning component of the controller, which is responsible for deciding on necessary actions to be applied to the SuOC and for increasing the quality of

this selection process over time. The learning component works on a defined set of possible actions and is not allowed to modify them or to generate new ones. In case there is no parameter set available suiting the current needs, an alternative strategy is needed. The architecture does not allow new rules (pairs of situation/conditions and parameters/actions) to be created randomly by e.g. Genetic Algorithms [122]. Instead, control is transferred to Layer 2 of the architecture.

Layer 2 of the architecture consists of a simulation tool and an optimisation technique and constitutes the "creative" component of the system. Based on observing current demands at Layer 1 (in cases of unknown situations or sub-optimal existing rules), the component has to find the best available parameter setting for the particular situation. If the action-selection process of Layer 1 has no matching knowledge or the existing actions have not been performant enough, the question "What would you do in this situation?" is given to Layer 2. Based on an optimisation heuristic, Layer 2 evolves parameter settings and repeatedly analyses them using a simulation tool. This bears the advantage that newly created parameter sets are not directly used in the live system, as this could cause the system to perform badly or even malfunction. Only those parameter sets qualifying in the simulator of Layer 2 are passed back to Layer 1, and may then be applied in the real world. Therefore, Layer 2 allows for a kind of "sandbox"-learning without the risk of applying arbitrary parameter sets to the live system.

On the highest (or *all-embracing*) level, the Layer 3 component is responsible for communication and collaboration among distributed autonomous elements – each system being equipped with an additional CM is able to communicate with others. Furthermore, this layer provides the interface to the user or administrator of the system. Hence, the user has access to current system measurements in order to monitor the system's performance. Additionally, he can change or adjust the goal of the system by adapting the objective function. Figure 3.3 depicts a network-wide view of the system. Several systems are equally wrapped into the proposed architecture. They act within the same environment and communicate via Layer 3. In addition, the user of the system can use the interfaces provided at Layer 3 of the particular autonomous elements to gain access to measurements, analysis, and the objective function.

The architectural design has some similarity with the *Viable System Model* (VSM) as introduced by *Stafford Beer* [123] as a recursively applicable template for structuring the management of large enterprises. *Beer* distinguishes between five functional types of subsystems:

- the productive (value-adding) type
- the coordinating (of type 1 subsystems) type
- the optimising type (w.r.t. the current resource utilisation)
- the planning type

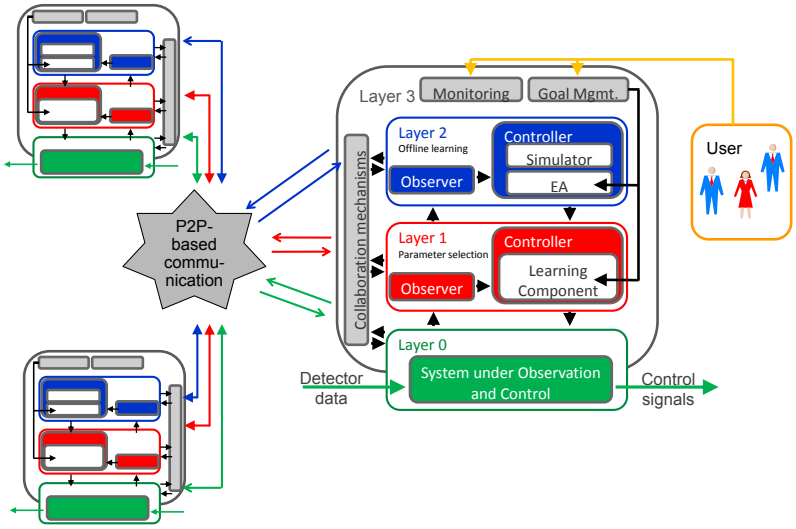


Figure 3.3: Network-wide view of the architecture

- the decision level type (defining the goal for lower types)

As depicted in Figure 3.2, the task assignments of the introduced Layers 0 to 2 are similar to those given by *Beer* with types 1 to 3. Type 4 is covered in a distributed manner by collaboration (Layer 3) among equal elements. Only type 5 is not explicitly foreseen and assumed to be situated in the user's area of responsibility. Although the VSM is explicitly non-technical[2], it can also be applied to various domains where autonomous entities form a collective organisation.

The remainder of this chapter will introduce the particular layers in more detail.

3.3.1 Layer 0: System under Observation and Control

On the lowest level of the architecture, the "productive" part (the SuOC) of the system is integrated into the proposed framework. Due to the targeted generic concept, the SuOC is not restricted to a particular application domain. Therefore, existing technical systems from varying domains can be controlled in the same way. Examples include urban traffic control systems (see Chapter 6), data communication protocols (see Chapter 7), or production systems (see Chapter 8.1).

Layer 0 encapsulates an existing technical system – consequently, both systems have to cooperate at runtime. In order to keep the effort for manually customising the existing system low, the Layer 0 component provides two basic interfaces: one for the observation of the SuOC and its surroundings and one for accessing the variable parameters. During

[2] *Beer's* intention has been to model business organisations – thus, the original model belongs to the domain of economic sciences.

the integration process of the system S into the framework, the only part affecting the logic and internal structure of S is to provide access for these two interfaces.

Although the approach of the proposed architecture aims at providing a generic solution, an engineer applying the framework to his specific technical system S has to adjust it in a few points and to supply additional information needed for the further process. For the bottom level (Layer 0), these tasks are closely connected to the two basic interfaces as introduced before. The first task corresponds to the observation interface of the Layer 0 component. This interface has to provide access to all attributes representing the current status of S, attributes quantifying the current performance of the system, and variables representing the perceived environment. The information obtained by these attributes serves as basis for the adaptation and optimisation processes at the higher layers of the architecture. Hence, a complete description of the particular *situation* is needed, which is done by defining the *observation model*. This *observation model* represents the decision which attributes are observed.

Typically, the attributes of the observation model represent measurements for the current status of S (e.g. system attributes, parameter configurations, disturbed sensors, etc.), but also environmental data like the distribution of neighbours has to be taken into account. The environmental data to be observed depends highly on the particular system S and its application domain. Considering as example urban traffic control systems, the most important attributes of the current situation are the traffic flows over the intersections on a turning basis – which can also be used as part of the performance metric on Layer 1. Considering data communication protocols, such a general statement is even more difficult, since network protocols vary in their purpose and application domain. Controlling mobile ad-hoc networks will typically adapt the parameter sets according to changes in the neighbourhood formed by other nodes – in contrast, Internet-based routers running the TCP/IP protocol stack will more likely be adapted according to observed latencies, delays, or traffic loads on specific links. Besides the pure information serving as condition for the adaptation process, the situation description has to provide all relevant information to analyse the performance of the adaptation process locally.

The second task considers the particular system and its adaptation at runtime – which parameters have to be changed according to the observations? A system to be controlled by the framework has to possess runtime-adaptable parameters with influence on the system's performance. The second task for an engineer applying the framework to a system S is to define the variable parameters of S that will be subject of control interactions by the Layer 1 component. Although these parameters are application-specific, they are typically similar within a domain of applications. For instance, these parameters might be durations of green times in traffic control systems; in data communication networks, such parameters are buffer sizes, interval lengths, counters, or delay times.

The lowest layer of the proposed architecture is characterised by the encapsulated "productive system". Based on the assumption of OC that a system has to remain fully operable

although disturbances might lead to failures of single components, the proposed layers of the architecture are strictly separated and operating autonomously. For instance, if the Layer 1 component fails, Layer 0 and its SuOC are not affected. As a consequence, they continue working on a static basis with the last parameter configuration – this will lead to a sub-optimal performance until Layer 1 is operable again, but at least the system works.

Besides these general characteristics of OC systems, Layer 0 incorporates further attributes regarding real-world applications. It is designed to be performed in real environments leading to possible influences that do not occur in systems sealed off from environmental impact. Following again the considerations by *Rao and Georgeff* [6] (cf. Chapter 3.2), Layer 0 is mostly characterised by noisy sensor values, possibly disturbed components, continuously changing conditions, and non-deterministic or non-predictable behaviour.

3.3.2 Layer 1: On-line Adaptation

Layer 1 of the framework provides the functionality to adapt the SuOC by actively changing its accessible parameters. Therefore, it contains two components to achieve the best possible adaptation strategy: an *observer* and a *controller* (see Figure 3.2). Figure 3.4 depicts the formation of a control loop consisting of these two components in combination with the SuOC at Layer 0. The control loop and thereby Layer 1 is processed in a discrete time interval, typically referred to as *sampling rate*.

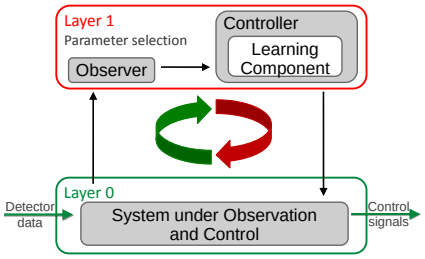


Figure 3.4: A first control loop defined by Layers 0 and 1

The observer is responsible for monitoring the SuOC and its surrounding environment. It receives a description of the current situation by querying the observation-interface of Layer 0. This situation description (cf. $y(t)$ in Chapter 3.1) contains data about the SuOC's settings, the status of the environment (if needed and accessible), and attributes to quantify the SuOC's performance. All of these values are based on sensor-data – thus, they might be noisy or subject to disturbances (i.e. malfunctions of sensors). Therefore, the observer serves as preprocessor: if possible, it filters bad values. Furthermore, it might contain a *prediction* component calculating a forecast for some attributes based on historical observations or existing knowledge (depending on the underlying technical domain, *approximation functions* might exist). Besides predicted and observed values, a sophisticated view on the measured attributes might need additional information about historical data. As a result, this processed and augmented data is transferred to the controller. Figure 3.5 depicts a detailed view of the observer component.

Based on the preprocessed data provided by the observer, the controller has to decide about necessary actions to be applied to the SuOC. This action-selection represents the

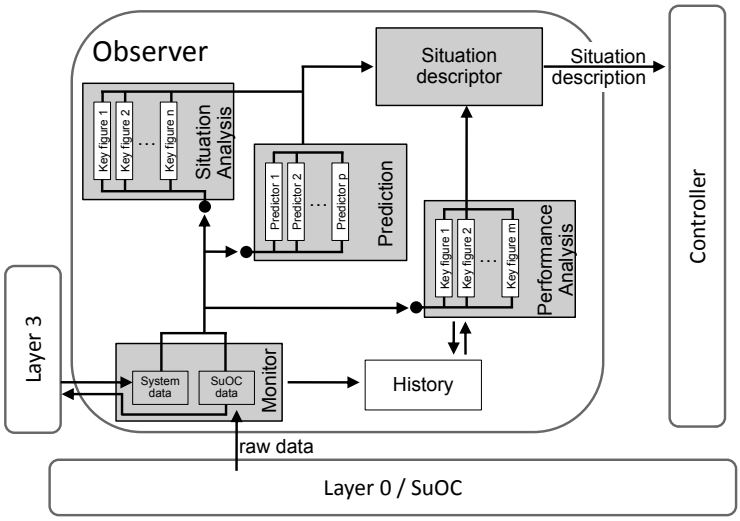


Figure 3.5: Detailed view of Layer 1's observer

problem to choose the best strategy for the current situation (select the best parameter configuration for the SuOC: $y(t) \rightarrow c(t)$). Thus, it is that part of the concept which initially enables adaptivity. The Layer 1 controller contains a set K of already known actions $K \subseteq CS$ to choose from in combination with knowledge about for which situation the particular actions have been chosen.

Due to safety reasons, the system is not allowed to try or use untested actions – it has to choose from the set of existing ones. In real-world applications, boundaries for the resulting behaviour of the technical system have to be guaranteed, since e.g. malfunctions are not tolerated by customers. Especially systems with a high degree of self-organisation and freedom to adjust their behaviours autonomously have to work within a tolerated corridor. As a result, the system's regulations have to be verifiable within given boundaries. In the context of this thesis, two main techniques are used to ensure the desired effect:

1. **Limited set of actions**: The set of selectable actions is fixed for the controller of Layer 1 – it is not able to create or modify actions. Actions are either initially inserted to the rule-base by an engineer or evolved at runtime using the Layer 2 component.

2. **Restricted choices**: The system is not allowed to choose any possible action from its rule base. Instead, it has to respect a *measurement of similarity* defining how similar an observed situation is to the one the particular action is dedicated for. The set of possible choices is then limited by a given threshold t_{min} defining the minimum allowed similarity (the maximum distance).

These limitations ensure the desired behaviour of the system. It is allowed to autonomously adjust the SuOC bounded by policies of the user. In addition to the automated adjustment of parameter configurations, the system is designed to improve the performance

of its selection process over time. In order to enable such a self-improving behaviour, the controller contains a *learning component*. The task of this component is to draw conclusions from the measured system performance and the recently performed control decisions. More formally, the system has to learn the mapping from situations $y(t)$ to actions $c(t)$ aided by an observable fitness measurement (quantified by f): $f(z(t)) \rightarrow \mathbb{R}$.

The situation descriptions as measured and provided by Layer 0 are based on attributes from a real-world environment. They are defined on a subset of those variables describing the system's particular surroundings. These attributes are characterised by typical characteristics of the real-world, the most prominent in this context is that they are continuous values in most cases. Due to the usage of continuous data to describe the current situation, the action-selection mechanism has to incorporate continuous values, too. This leads to the following problem. A situation will hardly appear again – expressed by exactly the same attributes contained in $y(t)$. The specificity of this statement depends on the size and the nature of the underlying search space and is therefore a question of the specific scenario the system has to work in. Consequently, the action-selection component is not able to store an exact mapping between all situations and actions. In contrast, it has to decide which from the known actions promises the highest benefit. This *highest benefit* cannot be pre-estimated or determined via deterministic techniques (like using always actions whose corresponding situation is most similar to the observed one in terms of a similarity metric), since the actual shape of the underlying fitness landscape is unknown. Thus, the system needs the possibility to determine the best match between situations and actions automatically at runtime. In other words, the system has to be able to *learn* this match – which is the task of the *learning component*.

Learning necessarily relies on the possibility to try different solutions and then determine which has been the best one based on a certain *reward* or *feedback*. This means that the learning mechanism takes over the responsibilities of the action-selection mechanism. Thus, the same restrictions have to be respected as defined for the action-selection. The learning component is not allowed to create actions on its own and it has to consider the similarity metric in combination with a predefined minimum of similarity. Chapter 4.1 discusses how automated learning under these restrictions can be realised.

After performing the necessary process, the controller might have chosen to adapt the SuOC. Therefore, it calls the adaptation-interface of Layer 0 and provides the new parameter configuration. Alternatively, it might keep the current configuration of the SuOC and abandon an adjustment. Furthermore, if the current situation is unknown or the learning component contains only rules with bad rating, it can trigger Layer 2 to find a new parameter setting for exactly the observed situation. Figure 3.6 depicts a detailed view of the controller component.

Although the Layer 1 component of the proposed architecture provides a generic approach, three main customisation tasks have to be fulfilled. Similar to the tasks at Layer 0, these tasks are needed to cover the specific character of the SuOC.

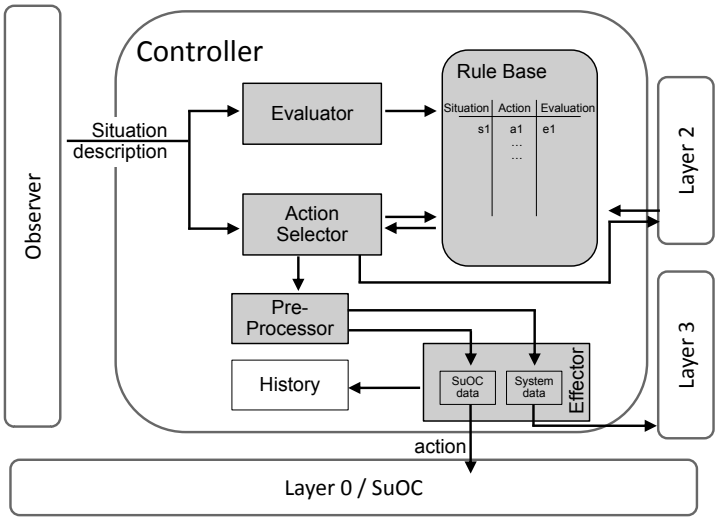


Figure 3.6: Detailed view of Layer 1's controller

(1) The first task is related to the automated learning. In order to equip the system with the possibility to analyse its own behaviour and improve it over time, an **evaluation measurement** is needed. An engineer customising the framework for the control of a new system has to assure the availability of necessary performance indicators when initially defining the situation description at Layer 0. Based on this data, a quantification method is needed that allows for a fast computation of a measurement for the system's performance within the last cycle. This evaluation measurement is also subject to control interactions of the user via the application interface provided at Layer 3. An external change of the evaluation incorporates the possible *flexible character* of technical systems (i.e. the setting of new goals).

(2) Besides the evaluation measurement, a **quantification of the similarity** between situation descriptions provided by the observer is needed. Due to the divergence induced by continuous values, the system needs a possibility to compare situations and compute their similarity. Therefore, the engineer has to define a formula to compare these situation objects.

(3) Finally, the observer can take **prediction** values into consideration when processing and augmenting the observed attributes in order to provide a complete situation description for the controller. Thus, an *optional* prediction model has to be defined. For some attributes, prediction models exist (e.g. for the prediction of movements of neighbouring nodes [124]) – they can be adapted for the corresponding attributes. Alternatively, these models have to be developed and integrated into the framework.

Similar to the encapsulation of the SuOC and its Layer 0, Layer 1 remains fully operable if higher layers fail. In this case, the adaptive control loop defined by the Layers 0 and 1 will still work as designed; but the absence of Layer 2 will lead to a static set of possible

actions, which restricts the behavioural repertoire. Furthermore, Layer 1 is characterised by its response time. Since the system has to work under real-world conditions, it has to be able to react immediately on observations. Thus, both – the observer and the controller part – cannot perform complex and time-consuming calculations. Especially a validation of possible actions cannot be performed within feasible boundaries. Due to these restrictions, the controller acts on pre-evaluated solutions – the safety consideration affects the set of possible actions by only allowing tested solutions.

Finally, the system's Layer 1 has to perform real-world learning. Thus, it is able to self-improve over the complete operation period. This learning affects the quality of the action-selection and results in a better system performance. Due to safety reasons, the learning mechanism works on a defined set of possible actions at the beginning. Afterwards, the behavioural repertoire is extended using the Layer 2 component and its sandbox-learning mechanism.

3.3.3 Layer 2: Off-line Learning

The Layer 2 component of the architecture is responsible for extending the behavioural repertoire of Layer 1 in case of unknown situations or insufficient knowledge. Thus, the component realises the *creative part* of the system by implementing the *sandbox-learning* principle. Such an off-line exploration allows to find appropriate actions without actually having to test different alternatives in the real world. The latter could be detrimental, as testing out potentially bad strategies in the real world can cause tremendous cost and cause the system to fail permanently. Hence, it is guaranteed that the *trial-and-error part* (which is necessary for unsupervised automated learning, see e.g. [125]) does not affect the performance of the SuOC.

The Layer 2 component works on-demand – every time the Layer 1 component does not know an adequate response to observed situations (e.g. no actions are known or only actions which proved to be unqualified), it creates an *optimisation task* for Layer 2. The observer of Layer 2 utilises the same pattern as discussed for the control loop at Layers 0 and 1 by monitoring the adaptation component of Layer 1. If an optimisation task occurs, it is passed to the controller, which is responsible for generating a new rule. In addition, the Layer 2 observer has to monitor the system resources and their utilisation by other tasks. Consider, for instance, a standard PC performing an additional Peer-to-Peer client to download large files from the overlay network (see Chapter 7.3.3). The generation of a new rule relies on an optimisation task, which is performed using simulations as metric. Both – optimisation (a large set of trials) and simulation – require high effort in terms of resource usage (CPU and RAM). During such a rule-generation process, load situations might appear where a PC will react slowly and does not provide the normal convenience for the user. Consequently, the protocol adaptation has impact on the user and is not transparent anymore – this impact

has to be avoided, otherwise acceptance problems will occur. Thus, the Layer 2 observer monitors (and maybe predicts) the resource utilisation – the controller is only activated during idle times.

The controller of Layer 2 is responsible for generating new rules. It receives the optimisation tasks and has to find the best possible action for these conditions. Therefore, it combines an optimisation heuristic [126] with a simulation component. The optimisation heuristic generates the particular candidate settings for the SuOC based on a target-oriented approach. The performance of the candidate within the simulator serves as measurement for its quality. Based on a "survival of the fittest" concept, the best setting is found after several iterations of the optimisation cycle. The architecture defines the need of such a technique, while Chapter 4.2 presents an investigation on which particular technique should be used. A model-based planning is always limited by the necessary simplifications made in the model. Thus, the best action with respect to the model is not necessarily the best action with respect to the real world. Therefore, Layer 1 is allowed to fine-tune the selection process by choosing between closely-situated conditions.

The second part of the controller is the *simulation tool*. Since nowadays many technical systems are tested and developed based on simulations before building them or applying them to the real world, simulation tools for nearly all significant technical solutions are available. For those cases where no professional tool exists, an abstract implementation of the problem and the system's behaviour using *Multi-Agent Toolkits* like *Repast* [127] or *Mason* [128] might be possible. Apart from the specific simulator, the purpose is to test the behaviour of the SuOC with the parameter set generated by the optimisation heuristic without applying it to the productive system S and the real environment.

The simulation tool itself has to be configured using a *simulation model*. This model defines what exactly has to be simulated. It specifies the simulated SuOC's status (e.g. topology, abilities, disturbed components, etc.) and the environmental conditions (e.g. neighbours, measured conditions). The situation description received from the observer has to be turned into such a model by considering the internal values. In addition, the simulated SuOC is configured using the parameter setting to be tested. For instance, the previously used example of urban traffic control can be considered as follows. The simulated intersection controller is configured with the control plan to be tested. Furthermore, the situation in urban road networks might be defined using the observed traffic flow (in $\frac{vehicles}{hour}$) for all turning movements crossing the intersection. Most importantly, the simulation relies on a realistic model of the intersection's topology. The simulated traffic is generated according to specific car-arrival-patterns from the field of traffic engineering. The amount of cars simulated for each turning represents the measured values from the situation description.

In contrast, the control of data communication protocols relies on other simulation tools. Their purpose is to represent real-world data communication with all possible effects (lost packets, sending distance in wireless networks, dispersion models, etc.). For instance, a simulation of a mobile ad-hoc network mainly relies on the protocol's state machine, a movement

model for the nodes, a message model, and a distribution of nodes within the simulated area. Thus, the situation description will cover an aggregated view of the neighbours' positions in order to find the best parameter setting for these conditions.

In order to enable the rule generation of Layer 2, one final task has to be fulfilled by the engineer. He has to provide the previously discussed simulation tool and the corresponding simulation model. The current framework has already integrated tools for traffic control (*Aimsun* [129]), data communication (NS-2 [130], Omnet++ [131]), and Multi-Agent Systems (Mason [128], Repast [127]).

The Layer 2 component of the proposed framework is responsible for the off-line generation of rules (mappings of situations to actions). In comparison to the Layer 1 component, it is characterised by a larger time horizon: no immediate answer is needed for a given stimulus. Thus, a possibly delayed response due to model-based evaluation is possible. This simulation-based learning represents the creative part of the whole system, since it is responsible for discovering new actions. The corresponding trial-and-error part when discovering new actions takes place in a "sandbox" – the *simulation tool*. Thus, the safety-critical part of the learning process does not influence the real-world system.

Besides the safety aspect, Layer 2 has another key-characteristic – the effort. In contrast to the relatively light-weight processes on Layers 0 and 1, simulation-based optimisation requires high computational effort mainly provoked by the simulation tool and the large number of simulations caused by the optimisation process. Although Layer 2 acts under a larger time horizon, it also dictates the speed of the self-optimisation – it remains the only possibility to extend the behavioural repertoire of Layer 1.

3.3.4 Layer 3: Regional Cooperation

The highest layer of the architecture (Layer 3) provides the basis for communication and collaboration between neighbouring systems. In contrast to the other three layers, the tasks and responsibilities of Layer 3 are not completely part of the focus of this thesis, since collaboration and cooperation is highly application-specific and cannot be covered by a generalised approach. Albeit, it provides basic key features of the system, which are discussed in the following. Additionally, some further application-specific mechanisms are outlined when discussing the particular applications, see Chapter 6.4 as example for the OTC system.

As depicted in Figure 3.2, Layer 3 is not designed following the concept as used for the three basic layers. Instead, it embraces the basic system (defined by Layers 0 to 2) by providing collaboration and communication capabilities for possibly all contained elements. Furthermore, it defines the interface to the user of the resulting system. User interaction will mostly follow two main targets; the corresponding entities are explained in the following part of this chapter:

- **a)** control and analyse the system's status and performance, and
- *b)* change or manage the system's goal, which allows for flexibility in terms of OC [16].

Connection to Layer 0: The connection between Layer 3 and Layer 0 aims at extending the perception area of the isolated entity (framework and SuOC). The basic system (Layers 0 to 2) is designed to operate on local information only, which might not reflect the complete available information at the particular node. To alleviate this restricted perception, the Layer 0 components of directly neighbouring systems can be allowed to exchange their sensor information through Layer 3. Considering the initially named example of urban traffic control at intersections, an exchange of sensor information between neighbouring intersection controllers can be used to improve the switching decisions of traffic-adaptive controllers. For instance, so-called *NEMA-controllers* [132] take gaps in arriving vehicle queues into account – closely situated intersections are able to determine the best gaps in the approaching traffic based on sensor data of their neighbours.

Connection to Layer 1: Besides a direct exchange of sensor data at Layer 0, communication on a higher abstraction level is performed between neighbouring Layer 1 components. This communication can affect both – the observer and the controller part of Layer 1. The observer is responsible for aggregating and augmenting the collected data from Layer 0 in order to provide a situation description that reflects the current status of the system and its perceivable environment. In this context, collaboration can provide a possibility for the observer of Layer 1 to build even more reliable situation descriptions. Considering the traffic control example, the observer can generate an improved prediction of the traffic situation by taking the neighbours' current situation into account, since this traffic will probably arrive within a given time interval at the corresponding incoming section. Such an improved prediction allows for an earlier adaptation of the parameter settings and consequently allows for further increasing the SuOC's performance.

In contrast, the controller part directly affects the adaptation strategy for the SuOC's parameter settings. Collaboration mechanisms can be applied to negotiate the particular configurations of some parameters. For instance, urban intersection controllers are typically coordinated along large traffic streams such as main arterial roads to form so-called *Progressive Signal Systems* (PSS). Therefore, the begin of a cycle is adjusted in such a way that arriving vehicles on coordinated streams do not have to wait at red traffic lights when arriving at the next intersection. This parameter defining the relative begin of a cycle is called *offset* (see Chapter 6.4.1) and can only be determined appropriately among neighbours.

Connection to Layer 2: The connection between Layer 2 and Layer 3 of the architecture affects the sandboxing. Simulation-based learning has the drawback of depending on the availability of computational power. Some scenarios will not allow for an on-demand usage of Layer 2 as originally implied by the architecture. For instance, a population of entities might contain a subset coming with less energy and computational power, e.g. nodes in wireless sensor networks (see Chapter 7.3.2). However, a situation-aware adaptation has

to be possible. Therefore, collaboration comes into play, which provides external Layer 2 capabilities by sharing those of neighbouring nodes. Beside this extreme example of a completely missing Layer 2, further even more popular possibilities are feasible. Neighbouring nodes can share their Layer 2 resources in case of idle times or return after failures. Furthermore, double creation of similar rules can be avoided by exchanging existing rules. Such a mechanism has been exemplarily investigated for the Organic Network Control System in Chapter 7.4.

Connection to the user: Finally, Layer 3 incorporates the flexible character of an OC system – the user or administrator has the opportunity to change the system's goal at runtime. Therefore, the *goal manager* provides access to the current objective function, which is considered by the learning component of Layer 1 when determining the reward for the last evaluation cycle and by the rule-generation component of Layer 2 when rating the performance of a candidate parameter set within the simulation tool. As example for such a change of the objective function serves again the traffic control domain. In high traffic periods, the predominant goal is to optimise the throughput. In contrast, the goal during low traffic periods might be adapted to decrease the delay times caused by red traffic lights. In addition, Layer 3 incorporates the functionality to monitor the controlled system and its behaviour – the user interface is situated here.

Due to the application-specific character of Layer 3, there are no well-defined tasks like for the other layers when applying the framework to a new system. But the system has to be able to communicate with its neighbours. Therefore, two basic components are necessary: a communication interface for physically exchanging messages between entities and an identifier for entities to recognise others. Within this thesis, these two components are assumed to be available.

The Layer 3 component provides a basis for on-line communication and collaboration between neighbouring systems. Since the purpose of collaboration mechanisms is application-specific, the component is mainly characterised by the application domain of the controlled SuOC. Correspondingly, the communication itself is characterised by the particular entities. For instance, communication in data communication networks as considered by ONC (see Chapter 7) can be established by taking advantage of the underlying communication channel of the SuOC. But even in such networks the communication differs: latencies and delays, link reliabilities, or delivery ratios of transmitted packages change with the type of the network. Similarly, each intersection controller in urban road networks is assumed to be connected with its direct neighbours, which reduces the problem to the one of ONC. But – in some cases – such an assumption might not be possible. Here, further technical equipment is needed or Layer 3 has to be deactivated.

3.4 Discussion of Requirements

Based on the previously presented architectural design of the framework, the next section discusses how the requirements as introduced in Chapter 2.1 are fulfilled:

A) Adaptivity: The control mechanism provided by the framework **enables adaptivity** for the parametrisable system. It adjusts variable parameters to changing internal and external conditions.

B) Robustness: The control mechanism provides **robustness against a set of disturbances**. In cases where the standard system would suffer due to unexpected behaviour, the adaptive version reacts and is able to find acceptable solutions within the granted degree of freedom. In this context, *disturbances* are not only failures and misbehaviour of individual components, but also unexpected situations where the static setup of parameters leads to a dissatisfying system performance.

C) No interference with the system's logic: The architectural design of the framework wraps existing parametrisable systems and provides OC-characteristics for static systems (Layer 0). The framework does not **interfere with the system's logic** – the communication is realised using well-defined observation and adaptation interfaces.

D) Operability: Each layer of the system encapsulates a specific part of the logic – the underlying SuOC is able to continue its work (**remain operable**) in cases where Layer 1 or 2 fail. It just falls back to a static character again by keeping the last configuration.

E) Flexibility: Besides disturbed situations, interventions of the user have to be covered. A possibility of **flexible goals** is foreseen allowing the user to exchange goals at runtime.

F) Vast situation and configuration spaces: The system is designed to deal with **vast situation and configuration spaces**. Due to the two-layered learning concept, it can appropriately react on **unanticipated situations**.

G) Self-improvement: Layer 1 contains a *learning* component responsible for **self-improving** the action-selection process at runtime.

H) Restricted exploration: The sandbox of Layer 2 and the boundaries for the learning component of Layer 1 **restrict the trial-parts** of learning – only pretested solutions are allowed. Since a pretesting at design time is infeasible due to the vast situation spaces, a sandbox solution is foreseen. This mechanism tests and evaluates possible actions for a given situation at runtime without interfering with the productive system.

I) Decentralised operation and collaboration: Layer 3 provides **decentralised collaboration** possibilities allowing for a controlled self-organised behaviour of several cooperating elements.

J) Comprehensibility: The system design provides interfaces for monitoring. In addition, the actions performed by the additional adaptation component can be traced and reconstructed – thus, the system's behaviour is **comprehensible to users**.

K) Real-world requirements: The control mechanism of the framework can deal with

real-world requirements, i.e. all kinds of improper inputs, noise, and continuous values.

L) Generalised approach: In general, the framework is applicable to all kinds of technical systems. Four examples are introduced in the remaining chapters. In order to investigate the general applicability, Chapter 8.3 provides a detailed analysis of this topic.

As a summary, the control mechanism provided by the proposed framework fulfils all defined requirements as introduced in Chapter 2.1.

3.5 Summary

This chapter discussed the target definition and the scope of the proposed system, followed by a detailed explanation of the architecture and its components. The resulting framework allows for on-line adaptation of an existing system S to changing environmental conditions. The architecture distinguishes between four different layers. The lowest layer (Layer 0) wraps an existing technical system S into the framework by providing access through two basic interfaces: one for the observation of the SuOC's and its environmental status and one for altering the parameter settings. In combination with the on-line adaptation component of Layer 1, Layer 0 forms a first control loop of the system. Based on automated learning, the system is able to improve its performance due to an optimised parameter-selection behaviour over time.

The developed system provides a self-optimising behaviour with respect to the parameter selection. Therefore, automated learning is used without the drawbacks of "normal" learning: *trial and error*. In order to guarantee that the system does *not* lead to failures affecting the system's performance significantly, two main differences to standard learning approaches have been developed. The first restriction affects the set of possible choices for the learning component – it is only allowed to select parameter sets that have been generated for similar situations. Secondly, the explorative part of the learning component is encapsulated and transferred to Layer 2 – this layer incorporates the sandbox-learning paradigm. Parameter settings are tested under simulated conditions before they are applied to the real system.

The basic system is defined by the Layers 0 to 2 and describes an isolated autonomously acting system. Through the additional Layer 3, communication among and collaboration between neighbouring systems is enabled. Collaboration is application-specific and ranges from exchanging sensor data to negotiating parameter settings. The presented framework is designed to provide a generalised solution. However, some basic tasks have to be fulfilled before the framework is able to control a new system S and thus adapt it to changes in the environmental conditions. Besides typical smaller implementation issues, these are mainly the following five tasks:

1. **Situation (observation model)**: All internal and environmental attributes having

impact on the selection process or the performance measurement need to be available through the observation interface of Layer 0.

2. **Similarity measurement**: The selection process of the learning component needs a measurement of similarity for the situation descriptions to be able to choose only nearby rules.

3. **Performance metric**: The self-optimising behaviour of the framework is based on the existence of a metric defining good and bad behaviour.

4. **Configuration space (variable parameters)**: Obviously, a framework that is designed to alter parameters of systems at runtime has to know which parameters have to be changed.

5. **Simulation model**: If not already contained in the framework, a simulation tool and a realistic simulation model are needed to enable Layer 2 sandboxing.

The next chapter investigates which techniques should be applied to the learning task at Layer 1 and the optimisation task at Layer 2.

Chapter 4

Design Choices

The previous chapter introduced the framework's architecture with a special focus on discussing the particular layers and their responsibilities. In addition, the tasks of the layers have been outlined by defining abstract responsibilities. In contrast to this abstract focus, this chapter investigates these particular tasks in detail and determines solutions for possible design choices. For instance, the machine learning part of the architecture's Layer 1 component is defined based on its responsibilities and the designated result. Therefore, a specific technique is needed, which is capable of fulfilling the corresponding tasks. Hence, the first part of this chapter analyses the learning task, identifies possible techniques, and discusses the usage of these different mechanisms. Based on an additional comparative study within one exemplary application scenario, the achieved results of using these candidate techniques are compared.

Considering the architecture as depicted in Figure 3.2, a design choice like the one for the learning problem is also situated at Layer 2: the *rule-generation* component. The architecture defines the need of an optimisation heuristic that is capable of finding the best possible parameter configuration for the SuOC in one particular situation. Much effort has been spent on investigating fast and accurate techniques to optimise a given function – thus, the problem is similar to the learning problem outlined before. Hence, the second part of this chapter analyses the optimisation task, identifies possible techniques, and discusses the usage of them in the context of the framework. Finally, both aspects (*learning* and *optimisation*) are consolidated and a recommendation on which combination promises the best results within the framework is given.

4.1 On-line Adaptation Using Machine Learning

In the first step, the question of how to realise the central learning component of Layer 1 is investigated. *Automated or machine learning* has been a research area focussing on different approaches and concepts to realise learning in technical systems for decades [133]. As a result, a set of different techniques has been developed from which each single technique is characterised by varying strengths and weaknesses. Thus, it is not the purpose of this thesis to extend the existing set of different approaches, but to identify and adapt the most promising ones according to the demands specified by the controller of Layer 1. Hence, this section investigates the learning component of Layer 1 in detail and determines the most promising way to realise the learning task. The investigation of the learning component builds on previous research presented in [134].

Initially, the term "machine learning" needs to be defined to specify the context within this thesis. Based on this definition, the special characteristics of the learning problem are determined. Afterwards, possible techniques capable of coping with the problem are identified and adjusted to meet the specific demands. Thereby, typical problems from the field of machine learning are considered as reference and basis for a comparison. To validate these analytical considerations, an empiric investigation has been performed that compares the identified candidate techniques in an exemplary setting.

4.1.1 Term Definition: Machine Learning

Machine Learning (ML) is a well-established research area and has its origins in *Artificial Intelligence* [135]. It forms a superordinate concept for all types of artificial knowledge generation in computer systems based on *experiences*. Typically, an artificial system learns by means of examples and is able to generalise them. The target is not to simply store mappings between an input i_j and an output o_k in the form of $i_j \rightarrow o_k$, but rather to "recognise" the *regularity* within the sample data. This is necessary, since such artificial systems aim at being capable of classifying and handling previously unknown or unforeseen data.

ML techniques have been successfully applied to various application areas. Currently, they are mainly used in domains like data analysis, data mining, and pattern recognition [133]. Nowadays, an increasing distribution can be observed in all kind of "intelligent" systems, which are able to adapt themselves according to changing conditions [136]. These obviously heterogeneous application domains have motivated researchers from different areas to develop definitions for the term *machine learning* (or the basic *learning problem*) suiting perfectly to their special demands and interests. Thus, a consideration of learning within the proposed framework has to start with determining a definition that clarifies the demands of the particular component. Therefore, the most important term mentioned in the previous paragraph is taken into account: *experience*. Scientists like *Alpaydim* [136] summarise ML

as meaning to program a computer in such a way that a given performance measurement is increased by taking example data and historical experiences into account. More formally, the definition given by *Mitchell* is used in the context of this thesis [125]:

Definition 4 (Learning Problem) *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Mapping this definition to the framework and the considered application scenarios, the common problem becomes visible. The traffic control system has to select the duration of green times at an intersection according to a given performance criterion (e.g. minimise delays at the intersection). Based on the expected reward determined by the simulator when generating the parameter set and experiences with the parameter set in previous cases, the learning component can increase its selection strategy of the signal plans. Furthermore, the control of data communication networks has to select network protocol configurations (like buffer sizes, delays, etc.) according to a particular performance criterion (e.g. decrease the overhead in mobile ad-hoc networks). As a summary, the problem defined by the framework's design is to increase the system's performance by means of some kind of performance criterion due to experiences – and accordingly an improvement of the parameter selection strategy over time.

4.1.2 Characteristics of the Learning Problem

In literature, ML techniques are usually applied to artificial scenarios like controlling an inverted pendulum [137], the approximation of a mathematical function [138], or choosing the movements of an agent on a grid (the *Woods*-scenario [139]). Further popular application scenarios with a more realistic background (as e.g. discussed by *Mitchell* [125]) include learning to recognise spoken words [140], to drive autonomous vehicles [141], to classify astronomical structures [142], or to play games like backgammon [143]. Although these examples represent a variety of different tasks for the responsible learning component, different ML techniques can be used to realise the learning part – meaning they are able to improve the behaviour of the automated system over time by taking experiences into account. In addition, each learning problem has its own characteristics leading to the observation that not each technique is applicable for each problem domain.

Consequently, the question arises how the learning problem of the Layer 1 component is characterised in contrast to the already investigated problems. Although some of the mentioned problems have a real-world background (e.g. recognise spoken words, drive autonomous vehicles), most of the problems in literature are artificial, whereas the framework of this thesis is designated to solve more complex real-world problems from different domains. Hence, the first main difference to most of the already investigated problems is given

by the typical **characteristics of real-world scenarios** like insufficient information, noise, need of fast reaction times, or disturbances.

Especially compared to the Woods-scenario or backgammon, the Layer 1 component is confronted with highly complex and vast configuration spaces for both – input and output variables. As a result, the system has to cover **larger spaces for actions (configurations) and situations**. Considering e.g. the control of data communication protocols, the situation (or condition) serving as input to the learning component can contain attributes like available neighbours, available resources, or movements. Neighbours in mobile ad-hoc networks can highly vary due to the context – the extreme values of the possible range are defined 1) by no neighbours (alone in the countryside) and 2) by several thousands (during a football match or a concert). For the control of urban traffic lights, values for delay times, queue sizes, and traffic flows for each contained turning movement will be considered to describe the situation – all these values are real values ranging again from zero to several thousands. In contrast, the most important situation characteristic for backgammon is the distribution of the gaming pieces along the 24 fields of the backgammon board. Similar observations can be made for the action part. In comparison, the Woods-scenario consists of $6,561$ different situations in its simplest setting and provides just four different actions for each agent.

The third main difference is the **missing final target state** for Layer 1's controller. For backgammon or the Woods-scenario, the decision whether the agent has been successful or not is simple: if it reached the target (Woods) or won the game (backgammon), it receives a reward. In contrast, the continuous control of traffic lights at urban intersections or of data communication protocols has no final target state. There is no single task to be completed (like to find the food or keep the pendulum upright). The approach is based on a single-step reward function, but the reward as measure for the quality of the last action is always noisy.

Closely connected to the problem of missing target states is the fourth difference: the **subsequent state is not deterministic**. In literature, the function $\delta : S \times A \rightarrow S$ is defined (with S the set of situations and A the set of possible actions) which results in the state s_{t+1}. For the Woods-scenario, the mapping is deterministic: if the agent moves to cell x, the agent will be at that cell in the next cycle. The same observation can be made for the backgammon player – after his move, the opponent can choose from a set of valid moves and adjust the setting of the game in a deterministic way. Considering the control of mobile ad-hoc network protocols, the effect is not that simple – the subsequent state is also influenced by joins/leaves of other nodes or collisions on the channel – but this information is transparent for the system. Similarly, the urban traffic control system can predict a change of the delays for turning movements caused by a change of the control strategy – but there are mutual influences with the selection strategies of its direct neighbours.

Besides the subsequent state, other attributes are non-deterministic compared to standard problems – the one with the highest influence is the **non-deterministic reward**. Considering again the previous examples, the contrast is clearly visible. Within the Woods-

scenario, the agent receives a fixed reward for finding the food. The same setup is used for winning the backgammon game or keeping the pendulum upright. Although the Layer 1 component adapts the parameter configuration, the impact on the reward cannot be predicted absolutely correctly in advance, since other influential factors cannot be measured. The action of the system might have been correct, but environmental factors can disturb the effect in a way that the reward leads to a bad classification of the action. As example, the Peer-to-Peer filesharing protocol "BitTorrent" can be controlled by the proposed system with the reward defined as using the achieved download rate of the client. This download rate is influenced by the configuration of the protocol parameters, but also by factors the agent cannot even observe. As one example, a high load on the underlying network caused by other applications can lead to a decrease of BitTorrent's download rate and consequently to a bad reward. Thus, the agent cannot assume that action a applied to situation s will result at each time t_i in reward $r(s, a)$; but rather the reward can be modelled as a randomised variable with the expected value $E[r(s, a)]$.

Probably all classical ML techniques are based on the existence of *experiences* E either at design time or at runtime. The performance of the learning rate depends strongly on the number of **learning cycles** – simplified, one can state that as more different experiences E are available the better is the learning performance of the system. In some cases, this effect can turn in the opposite direction (see "overfitting" problem in Neural Networks [144]). Thus, the approach relies on a large number of learning cycles. In contrast, real-world applications have the problem that only few evaluation cycles are possible – in contrast to simulated environments where all possible stimuli can be generated and analysed extremely fast. Thus, the number of evaluations is restricted by the normal time of the day. Furthermore, the most prominent situations will account for the largest fraction of situations, while other situations will only occur seldom (and hence are evaluated once in a while). A system might be trained in advance, but the problem of on-line learning is still present, since only a small fraction can be considered at design time.

As a conclusion, we can state that the learning problem of the Layer 1 component has its own characteristics compared to standard learning problems. Hence, there is no commonly agreed standard solution from literature to be obviously applied. Thus, the following part focuses on determining candidate techniques and adapting them for an application in the framework.

4.1.3 Machine Learning Techniques for Layer 1

Based on the preceding consideration of the learning problem's characteristics, the next part of this section deals with the question of how learning has been covered by research and how these results can be applied to the particular problem defined for Layer 1. In addition, the question arises which of the existing concepts from literature are applicable and how they

have to be adapted to match the initially formulated safety restrictions.

Therefore, a first step is to classify existing approaches of machine learning to create a basis for the succeeding question of which techniques are promising candidates. Several different classifications are known in research, one of the most popular ones is based on the way knowledge exists in the particular learning techniques: *symbolic systems* (propositional logic, predicate logic) where knowledge is represented explicitly by examples and induced rules are distinguished from *subsymbolic systems* like Artificial Neural Networks (ANN), which are trained to show a specific behaviour, but which do not provide insights into the "learned" solution process – here, knowledge is implicitly represented.

Based on Definition 4, one more popular aspect to distinguish between different classes of ML techniques is given by the question of how the *experiences* E are present. Consider, for example, a computer program controlling a player in the game *four in a row*. One possible approach might be to provide a trainer that determines the ideal moves for specific situations. Alternatively, the experiences E can be present in a more indirect form – the automated player receives a feedback at the end of the game. Based on the outcome, it can analyse whether its sequence of moves has been successful or not. The two possibilities are representatives for two more generalised learning concepts: *Supervised Learning* realises the learning (or training of the system) by providing examples and *Reinforcement Learning* by means of reward or punishment (reflecting a positive or negative outcome).

Supervised Learning (SL) assumes to have an appropriate set of input and (correct) output pairs available at design time. Typically, techniques from the field of SL are applied to classify large data sets where an expert-classified subset already exists (see e.g. *LeCun et al.*'s detection of handwritten digits [145]). Thus, a prerequisite for applying SL to the learning task of the framework is the availability of optimal actions for different kinds of situations – ideally covering the complete situation space. What seems to be manageable for handwritten digits, is dramatically more complex for the control of varying technical systems. For handwritten digits, a validation is based on a configuration space of 10 different possibilities (the digits "0" to "9"). In contrast, a validation of correct matchings within the proposed framework depends on an optimisation component and the possibility to oversee the complete situation space at design time. Although the former aspect is potentially feasible, it is time-intensive and requires large computational effort. Even more limiting is the latter aspect. Typically, the situation space is too vast. In addition, a trained SL-system is not able to take a feedback of the environment into account and is consequently not able to further improve at runtime (or react appropriately to unanticipated situations).

The most famous representatives of SL are *Concept Learning* [146], *Decision Trees* [125], *Bayesian Belief Networks* [147], or *Artificial Neural Networks* [148]. Although they describe different approaches and even different types of representing knowledge (implicitly vs. explicitly), they all are based on a pretraining at design time and a succeeding static configuration at runtime. Thus, SL techniques are not applicable to the framework.

Reinforcement Learning (RL) differs fundamentally from SL. Although the system

also learns mappings between situations and actions, it is not supported by an external coach. There is no training set allowing to derive a policy or generalise inherent correlations. In contrast, the learning approach depends on receiving a *feedback* of the performed actions providing the possibility to evaluate its decisions automatically. This evaluation is done by receiving a numerical *reward* quantifying good or bad performance. If the initial state has been improved by the last action (or set of actions) considering the particular task, the system receives a high reward. In contrast, the system receives a low or even a negative reward, if the chosen action has proven to be ineffective or bad [133]. Since the SuOC's performance needs to be quantifiable using numerical values, the basic concept suits the initially defined requirements for learning in the developed framework.

Due to the *on-line* learning capability, RL is a widely investigated domain resulting in various different solutions. Within this domain, three major concepts have to be distinguished: *Temporal Difference* (TD) learning [143], *Q-Learning* [149], and *Rule-based learning* [133]. TD Learning has been introduced by Samuel in 1959 [150]. He investigated possibilities to develop an automated player with learning capabilities for the board game *Draughts*. The concept relies on an evaluation function for all states of the board – such a function estimates the probability that the own player wins. The estimations for succeeding states serve as training input for preceding states [151]. This simple concept is also the main problem for the application to the framework's learning problem. The next situation is predicted based on the historical ordering of situations – this assumes a strong dependency between situations and neglects unanticipated events like disturbances.

Q-Learning is an extension of TD and has been initially described by Watkins in [149]. He adapted the concept by changing the purpose of the evaluation function – it describes pairs of states and actions. The system learns a function Q mapping a pair of situation and action onto a numeric reward:

$$Q \ : \ S \ x \ A \ \rightarrow \ \mathbb{R} \tag{4.1}$$

If the system considers the expected reward as selection criterion, it has to determine *that* action maximising Q for the given situation. In classic Q-Learning [149], this selection process is done using a look-up table storing all pairs of situations and actions. Such a look-up table is not feasible for continuous and vast situation spaces. Beyond look-up tables, TD learning has been equipped with different update prediction functions (like function approximation) all trying to reduce the limiting effect defined by look-up tables. But even these variants have major drawbacks. They are not concordant with the safety-restrictions of Layer 1's learning component and there is no possibility to combine the approach with Layer 2's sandboxing.

Based on the early TD and Q-Learning approaches, *Rule-based learning* emerged as a further reinforcement-based method. Early work by Holland [152, 153] served as basis for further developments, leading to one of the most active and well-documented areas

in *genetic-based machine learning* [154]. As a result, *Learning Classifier Systems* (LCS) [155] and the closely connected *Fuzzy Classifier Systems* [156] are well-established concepts. Classifier systems store knowledge explicitly by using rules (the so-called *classifiers*), which allows for a comprehensibility of their behaviour. They learn on-line by receiving a reward from the environment and updating the performance values assigned to the classifiers. Classifier systems fulfil most of the requirements formulated for the learning task of Layer 1; the most significant deviation can be observed when considering the rule-creation process and the selection-base of the system. Thus, they are modified in the remainder to fully match the demands.

Besides the two popular concepts SL and RL, several other directions of research are known in literature. But these concept are neglected for varying reasons. In some cases, mathematical inference and reasoning is used, which cannot be applied to the problem due to inappropriate information and the lack of a consistent world model. In other techniques, knowledge about the fitness landscape defined by the learning problem is required (see e.g. [125, 136] for the previous examples). The discussion named only a few criteria why some concepts are not applicable and others serve as basis for further adjustments. Thus, it summarises the investigation performed on the basis of the initially formulated requirements for the learning component and the special characteristics of the learning problem as introduced previously. As a result of this process, a matrix has been generated classifying each considered learning technique according to these criteria – this matrix can be found in Appendix A.

4.1.4 A Modified Real-valued Learning Classifier System

As stated before, the most promising candidates are rule-based systems. Since there is no adequate rule set in advance and due to the need of an improvement over time, simple rule systems mapping a situation to an action are not useful. Instead, learning rule-based systems are needed. The most prominent representatives are Learning Classifier Systems (LCS). The field of LCSs as introduced by *Holland* in the 1970ies [152, 153] is probably one of the best-investigated genetic-based machine learning domains in literature [154] and is founded on the concepts of reinforcement learning. Holland describes the basic approach as a three-step process:

- classify the input,
- choose an appropriate action, and
- gain experience from observing the behaviour [155].

This reflects exactly the learning problem outlined by the architecture at Layer 1.

In literature, two broad categories of LCS are known: *Pittsburgh* and *Michigan* style classifier systems. The former one is based on concepts proposed by researchers at the

University of Pittsburgh [157, 158]. This type of LCSs focuses on learning as an off-line optimisation process. In contrast, researchers at the University of Michigan investigated on-line learning systems. Since an on-line learning is targeted here, Pittsburgh-style is neglected and Michigan-style is meant when discussing LCS in the following. Considering the Michigan-direction in LCS research, the *eXtended Classifier System* (XCS) as proposed by *Wilson* [159] is widely accepted as one of the most prominent variants and serves as basis for several extensions. Figure 4.1 describes the typical schematic design of an XCS – the basic concept is explained in the following, while the reader is referred to the literature for further details and a detailed algorithmic description [160, 161, 162].

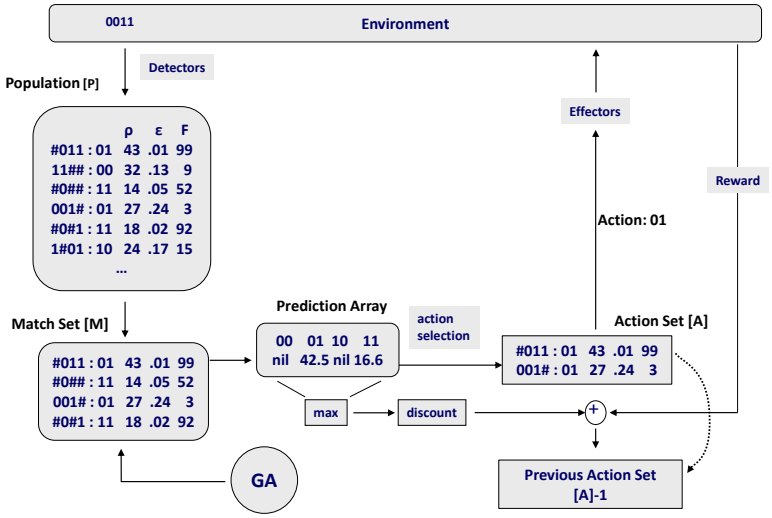


Figure 4.1: Schematic overview of an XCS according to *Wilson* [159]

In general, an LCS is designed to learn the best action for a given situation – which is typically defined as a vector of numerical values. In this context, *best* specifies that action resulting in the highest reward. Therefore, the system contains a set of possible actions from which it can choose the most promising one. The quality of this selection process is improved over time by taking the reward into account – this value represents the experience as mentioned in Definition 4. In other words, the environment provides some kind of *reinforcement* (also called *payoff*) serving as a measure for the quality of the last actions performed by the LCS. Considering the application within the proposed framework, the success of the SuOC's adaptation process is quantified by some kind of performance measurement for the SuOC. Based on such a performance measurement, the LCS can determine whether it is promising to choose a specific action again for a given situation or not.

In order to fulfil the described task, the LCS operates on large sets of rules, so-called *classifiers*. Each of these classifiers represents a mapping of a condition (the situation) to an action (a parameter set of the SuOC). Furthermore, it contains some rating attributes – the learning is done by updating these rating attributes according to the performance of the classifier when it has been applied to the system. A classifier within an XCS-style LCS consists of several more attributes (see Figure 4.1) [160, 161, 162]:

- **Condition**: $c \in C$ specifies the condition part of the classifier – the situation for which the classifier can be applied. C is defined as the complete situation space and corresponds to the set Z in Chapter 3.
- **Action**: $a \in A$ specifies what the system does, if the classifier is chosen in the particular situation. A correpsonds to the set CS as introduced in Chapter 3.
- **Prediction**: The prediction ρ is a forecast of the reward r, which is expected if the proposed action a_i is applied in the given situation c_j.
- **Error**: The error ϵ is a measurement of the wrong predictions occurred within the preceding evaluation cycles.
- **Fitness**: The fitness F reflects the reliability of the classifier's prediction.

These five attributes define the core of a classifier. In addition, further values can be found in different implementations, like the number of occurrences of the classifier in action sets, the averaged size of these action sets, or the last time the classifier participated in forming offspring. For the context of this thesis, these further attributes are irrelevant.

As depicted in Figure 4.1, the basic cycle of the XCS can be divided into two phases: the *performance* and the *evaluation* (or learning) phase. The former one determines one action and applies this to the live systems, while the latter one is used to learn from the achieved experience by observing the corresponding effect in the controlled system.

Performance phase: Initially, the LCS contains a basic set of classifiers – the *population* denoted as $[P]$. The *performance* phase begins with comparing the condition parts of all contained classifiers to the stimulus as observed from the environment – the *matching* classifiers become part of the *match set* $[M]$. Typically, $[M]$ consists of several different actions proposed by the contained classifiers, but the system needs only one. Consequently, it has to choose the most promising one. Thus, the next step is to build a *prediction array* $[PA]$ assigning one real-valued measurement to each distinct action proposed by the classifiers in $[M]$. In *Wilson's* XCS, this value is calculated as fitness-weighted prediction. As a result, each action in $[M]$ is rated by a numerical value – based on the corresponding distribution, the required action is chosen. Following these measurements for all actions in $[M]$, there are several different strategies known to choose the required action, which will be applied to the controlled system. For instance, one can obviously always choose the most promising one (highest-weighted prediction value), or rely on probability-based approaches.

Typically, a roulette-wheel-based determination process can be found in scenarios from literature [159], since this allows for a good trade-off between using the most promising action and allowing others to proof their performance and consequently gain experience. All classifiers proposing the selected action form the *action set* $[A]$. The corresponding action is performed afterwards and $[A]$ is stored for later updating within the evaluation cycle.

Evaluation phase: The second phase – the *evaluation phase* – is responsible for updating the evaluation figures of the classifiers. This part of the process derives the knowledge and stores it for further use. Since an XCS is an accuracy-based classifier system, the target is to increase this accuracy κ over time by using Temporal Difference Learning techniques [149]. After applying the selected action a_j to the system, the performance is observed and some amount of *payoff* is received. Based on this *payoff*, the performance values of all classifiers contained in $[A]$ are updated according to the observation. Initially, the prediction ρ of each classifier is altered in the direction of the payoff using a Q-Learning-like algorithm. The approach takes the maximum figure as part of the prediction array into account and discounts this by a factor (by multiplication).

In addition, each classifier's (contained in $[A]$) prediction error ϵ and fitness f are updated using the received payoff. Since the prediction ρ is a forecast of how the system will perform using this classifier in the matching situation, the prediction error ϵ describes the mean error of this prediction. In contrast, the fitness value f considers the accuracy of the classifier's prediction – it is computed as follows. Initially, the accuracies κ of all classifiers in $[A]$ are computed. Afterwards, the relative accuracy of each classifier is determined (divide κ_j by the sum of all $\kappa_i \in [A]$). Finally, the fitness value f is determined based on this relative accuracy, which makes f representing the accuracy of the particular classifier in relation to the accuracies of those classifiers typically being part of the same action sets. Consequently, a pressure is put onto classifiers to perform better than similar ones – according to the principle "survival of the fittest". More details of the update mechanisms for ρ, ϵ, and f are given by *Wilson* [159].

Besides these two main phases of an LCS, an important question arises: where do the classifiers come from? The answer to this question is founded on two different mechanisms: 1) the covering process and 2) the Genetic Algorithm (GA). If no matching classifier was found during the building process of the match set, a classifier consisting of a condition matching the current stimulus, a random action, and a default value is added to the rule base. This process is called *covering*. Additionally, a reproduction cycle is started sporadically. Within this cycle some classifiers are chosen to be "parent" individuals, and the genetic operators *crossover* and/or *mutation* are applied to copies of these parents to form *offspring*, which are inserted to the rule base [159]. Classifiers with high accuracy values are reproduced more frequently than less accurate classifiers. In *Wilson's* XCS algorithm, the generation of offspring is performed on the action set, other variants consider the match set or the whole population. For further details on this topic, the reader is referred to *Wilson* [139, 159].

Since the concept contains a mechanism to create new classifiers, it needs another mechanism to delete bad classifiers in order to keep the size of the rule-base at a manageable level. The complexity of an LCS depends on the number of contained classifiers. In each performance phase, the population has to be searched for matching classifiers. Afterwards, only subsets of the population ($[A]$ and $[M]$) are considered. The corresponding complexity of the LCS is $O(n)$ with n denoting the number of classifiers. In order to keep the search feasible although new classifiers are created continuously, a mechanism to get rid of bad or redundant ones operates on the population regularly. *Wilson's* XCS is configured to keep the population size at a given level. Therefore, the proposed concept selects existing classifiers to be substituted stochastically whenever new classifiers are created.

Modifications of the original algorithm: Although LCSs are evolutionary on-line learning systems, some modifications are necessary before using them for the control task of the framework. The modifications described in the following part of this chapter are based on previous work (cf. [163, 164, 2]). Existing systems like XCS (which serves as basis for a modified version) create classifiers in a stochastic process and evaluate their quality by applying their actions directly to the environment. For the task defined by Layer 1 of the architecture, this is inadequate due to, for instance, the safety requirements formulated before. In order to use an LCS within the proposed framework, both aspects of the classifier generation have to be adapted: 1) the covering mechanism and 2) the rule generation using GAs. The latter part is already considered within the architecture of the framework – instead of using a GA on the action set $[AS]$, new classifiers – or more precisely their action parts containing the SuOC's parameters – are evolved by an optimisation heuristic, which uses a simulation software to evaluate the parameters' quality with respect to a specific situation. Due to this *sandboxing*, an approximate quality of a classifier is known even if it has not been previously applied to the SuOC in the real environment. Although it is assumed that the simulation models of Layer 2 reflect the reality, imprecisions induced by the simulation-based evaluation are inevitable – however, the on-line learning performed by the LCS is intended to cope with these limitations. During the overall operation time, underperforming classifiers are sorted out – with *underperform* meaning that they do not reach the best performance, but still do not behave badly.

The concept of the previous paragraph demands that Layer 2 is activated whenever the LCS does not contain a matching or only inappropriate rules. Unfortunately, evolving good parameters based on simulations takes time, while an LCS is expected to react on changes in the observed environment immediately. Hence, a mechanism is needed that provides a promising action, although the rule base does not cover the current stimulus – in *Wilson's* XCS, this is done by the *covering* mechanism. Again, this part has to be adapted, since a randomised approach cannot be tolerated. Thus, covering is also customised by 1) selecting a classifier located most "closely" to the unmatched situation and 2) widening its condition as far as necessary to match the situation. The *widening* of existing classifiers located closely to an unmatched situation enables an immediate response of the LCS while, on the other

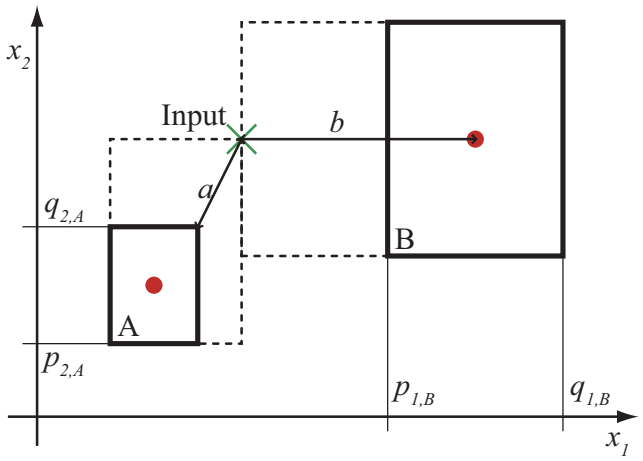


Figure 4.2: Modified covering mechanism

hand, the situation-dependent quality of the parameter set remains (somewhat) predictable. Figure 4.2 illustrates the concept of the modified covering mechanism using an example with a two-dimensional situation space. The point marked "Input" in the two-dimensional space describes the observed situation. In the example, the rule base contains only two classifiers: A and B. Both classifiers cover a certain part of the search space defined by intervals in each dimension. By copying A and B and widening their condition parts (the intervals) as far as necessary, new classifiers are created that match the current stimulus. Based on the discussed adaptations, the developed variant of *Wilson's* XCS can be applied to the framework and used as learning component.

Management of the rule-base: In addition, stochastically organised deletion of classifiers is not useful if the generation is done using time- and resource-expensive optimisations. Thus, the concept has been adapted as follows. Whenever new classifiers are created by Layer 2, the mechanism searches for other classifiers covering the same condition and exchanges them by the new one. Typically, this happens if the covering component has copied a nearby classifier. If no preceding classifier is found, a classifier is only exchanged when matching the following conditions: 1) it has a given minimum experience (has been part of action sets), 2) a relatively low fitness value (compared to the rest of the population), and 3) it has been created by covering. Layer 2-based classifiers are assumed to be optimal and typically covering-based classifiers make up for the largest part of the population – thus, they are preferably chosen.

Flexibility: Furthermore, XCS (like other LCS variants) does not cover *flexibility* issues. A fixed fitness function is used considering the payoff and consequently updating the performance values ρ, ϵ, and f of the classifiers. Currently, the approach has been extended by storing such a triple (ρ, ϵ, and f) for *each foreseen evaluation function* of the system. This allows the user to switch on-line between a set of given possibilities. Current research focuses

on more appropriate solutions [165]. Besides these previously explained adjustments, the rest of the XCS algorithm as proposed by *Wilson* [159] remains untouched. The presented variant has been implemented based on the reference implementation and its configuration as presented in [161].

4.1.5 A Modified Real-valued Fuzzy Classifier System

Traditional LCSs as discussed before are designed to learn matching actions for situations that can be represented as character strings [166, 139, 159]. This representation is not applicable to most of the real-world learning problems, because sensor data (which is the basis of the situation) is typically characterised by *continuous values* and *error-prone*. *Wilson* (as well as the adapted XCS variant presented before) deals with the problem by modelling the condition part of the classifiers using intervals [167]. An alternative concept is formulated by **Fuzzy Classifier Systems** (FCS, [168]) – it is inspired by the way humans percept their environment. Typically, humans describe their perceptions by using qualitative expressions like *large* and *small* or *hot* and *cold*. This classification is not based on objectively measurable boundaries – instead, they are *subjective* and *fuzzy* [169]. Classifiers for FCS map such a fuzzy classification onto the condition part by introducing *linguistic terms*. In addition, the continuous *membership function* $\mu(x)$ is needed quantifying the degree of membership of element x in a linguistic term. Figure 4.3 illustrates such a membership function for the example of humans describing the temperature. There are fuzzy ranges where a given temperature belongs to two linguistic terms (e.g. 25° belongs to "cold" and "warm" with 50 % degree of membership each). In order to enable this representation within FCS, *Casillas et al.* [138] propose to use the disjunctive normal form (DNF) as follows:

$$IF\ ((X_1\ is\ \tilde{A}_1)\ and\ \ldots\ and\ (X_n\ is\ \tilde{A}_n))\ THEN\ (Y\ is\ B) \tag{4.2}$$

where each input variable X_i represents a set of linguistic terms: $\tilde{A}_i = \{A_{i1} \vee \ \ldots \ \vee A_{ij}\}$. The output variable Y is a single usual linguistic variable. In the original concept, the classifiers have been encoded using the following scheme. Let (S [small], M [medium], and L [large]) be the three linguistic terms for each input/output variable used in the example, then the fuzzy rule

$$[IF(X_1\ is\ S)\ and\ (X_2\ is\ M\ or\ L)]\ THEN\ (Y_1\ is\ M)\ and\ (Y_2\ is\ L) \tag{4.3}$$

is encoded as [100|011||23].

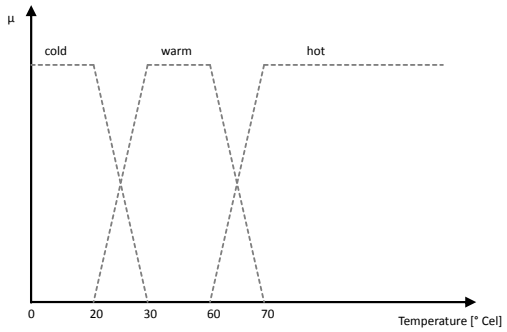


Figure 4.3: Fuzzy Sets and memberships

Figure 4.4 describes the basic concept of an FCS, which has already been applied to various problems – like the control of autonomous agents [170, 171] or cancer diagnosis [172]. The matching is based on *fuzzy sets* and results in building the match set $[M]$. Afterwards, several non-redundant and consistent *candidate subsets* $[CS]$ are generated and the action set $[A]$ is chosen out of them. Finally, the *fuzzy interference* is used to aggregate the actions contained in $[A]$ into one action a_i that is applied to the system via the effectors. Due to the fuzzy concept, some modifications of the original XCS algorithm are required as discussed by *Casillas et al.* [156, 138]. The following part describes the differences to the basic XCS algorithm:

- **Match Set**: The match set $[M]$ is constructed by all classifiers with a matching degree greater than zero for the current input.
- **Prediction Array**: In FCS, classifiers do not support one single action. In contrast, they support different ones with a certain degree of membership. Therefore, the $[PA]$ is replaced by the computation of candidate subsets ($[CS]$). Each of these $[CS]$ is computed by grouping all classifiers matching the input into different groups of consistent and non-redundant fuzzy rules S_i with the maximum number of rules in each group. *Casillas et al.* [138] use a greedy-based heuristic to generate these groups.
- **Action Set**: The previous step resulted in several different candidate subsets. From this set, one candidate set is selected as Action Set: $[A]$ is that rule subset from the set of $[CS]$ with the highest mean prediction.
- **Effectors**: In XCS, only one action is supported by $[A]$. In contrast, $[A]$ in FCS supports different actions each with a certain degree. Thus, the final action has to be derived from $[A]$ – the approach is based on calculating the centre-of-gravity of the parameter space defined by the contained classifiers. As a result of this so-called *defuzzification*, one action is derived, where each single parameter is calculated from the set of supported actions. Hence, the final action of the FCS may not be contained in the population. On the one hand, this contradicts the to safety requirements, but, on the other hand, a similarity of nearby situated parameter configurations is assumed to alleviate the apprehended effect.

Similarly to the XCS approach, the basic algorithm has been adapted to match the safety demands of the learning component:

1. **Covering mechanism**: Inspired by the standard XCS algorithm, *Casillas et al.* propose the selection of a random action as covering mechanism [138]. This would again lead to undesired behaviour in the proposed framework. Thus, the covering mechanism is reduced to select nearby rules (based on a certain measurement of similarity) and the usage of a given standard parameter set as discussed for the XCS.

2. **Rule generation**: Within the FCS concept, a GA is responsible for creating new rules. This capability is taken over by Layer 2 in the same way as described for the adapted XCS variant.

3. **Membership functions**: The fuzzy approach relies on the existence of fuzzy sets defined for the configuration space of both – input and output variables (situation and action part). Thus, the user has to decide about the number and the form of their membership functions. Since the form of the fuzzy sets has typically no relevant impact on the performance of the system [173], the adapted variant relies on easy to compute forms in terms of determining the centre of gravity: *isosceles triangles*.

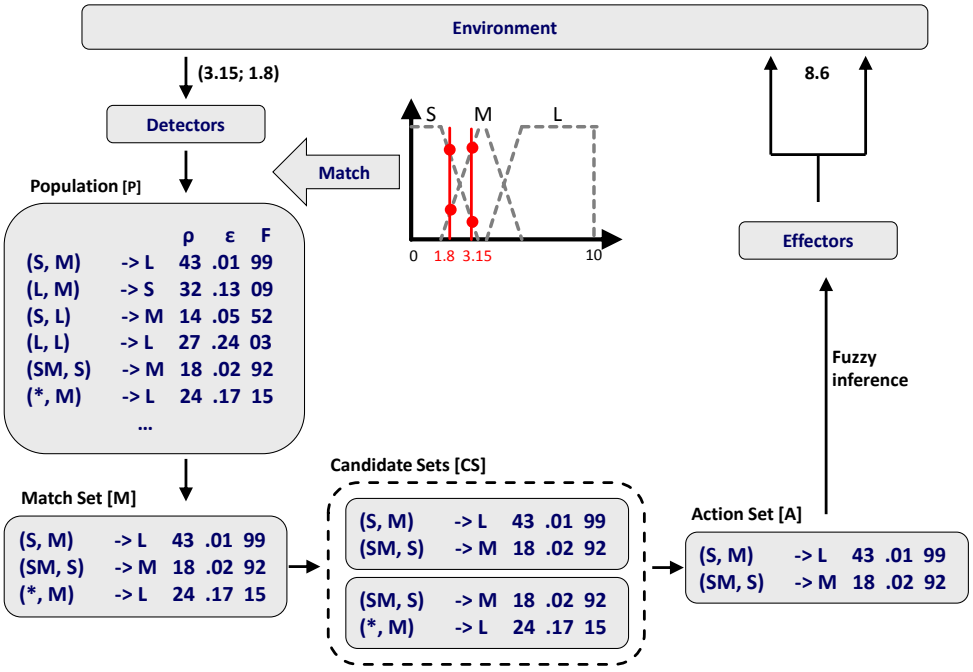


Figure 4.4: Concept of a Fuzzy Classifier System according to *Casillas et al.* [156]

Based on these adaptations, a variant of FCS to be applied to the learning task of Layer 1 has been developed. Rules are only generated by Layer 2 using the sandboxing approach

and the covering mechanism is again defined on the similarity metric, which allows only "nearby" rules to be used. Although some modifications have been performed, the original algorithm of FCS remains untouched.

4.1.6 Comparison of Learning Techniques

Previously, two machine learning techniques have been identified as promising candidates. The following part of this chapter applies these techniques to a representative scenario for the framework and compares the received performance. In order to extend the basis of comparison, ANN have been applied to the problem, although the approach has been sorted out due to (among other criteria) the missing concordance to the safety restrictions. The desired effect of taking ANN into account is to demonstrate that the on-line learning techniques outperform the off-line learning techniques. The candidates *LCS* and *FCS* are considered based on the adapted versions introduced before. Details on the adaptation of the ANN-variant are given in [134] as well as a further comparison taking a combined variant of Q-Learning and ANN into account.

Chapter 8.3 investigates an abstracted model of the underlying learning problem and analyses for which type of problems the developed framework is applicable. Therefore, a classification according to four different types of problems is defined. The evaluation of the abstracted model results in the insight that systems characterised by problems of type 1 or type 2 can be controlled by the developed framework. In addition, those of type 3 can be controlled under certain restrictions. The OTC and ONC scenarios presented in the remainder of this thesis can be classified as type 2. Since systems belonging to the same type of learning problem are exposed to the same characteristics of learning, it is assumed that one of them can be considered exemplarily for the complete set.

As basis for the empiric comparison serves an exemplary application of the proposed framework: the Organic Network Control System (ONC). ONC has been developed to automatically adapt network protocol parameters according to changes in the environmental conditions. For details of ONC, the reader is referred to Chapter 7. As one example, ONC has been applied to the domain of Peer-to-Peer protocols. In particular, a single instance of the filesharing protocol *BitTorrent* [174] serves as SuOC that has to be adapted on-line.

The comparison has been performed using the BitTorrent implementation for the standard network simulation tool NS-2 [130] as developed by *Eger et al.* [175]. Four consecutive downloads with one hour download duration each serve as basis for the simulation. Due to comparison reasons, the characterising aspects of the downloads (especially the ordering or the sizes) are given in advance, which makes them identically repeatable. The learning component is triggered in discrete time steps of one simulated minute and has to decide whether an adaptation of the protocol's parameter configuration is necessary or not. Thus, each simulation contains 240 discrete evaluation cycles for the learning components. The

presented results are averaged figures taking 50 simulations into account.

The performances of the learning components are compared under exactly the same conditions to the standard configuration and to each other. Therefore, the goal of the learning component is to increase the achieved download rate of the BitTorrent protocol. The timing of the simulation for all peers and seeds is depicted in Figure 4.5. In addition, the figure describes the availability of download bandwidth – the utilisation of this available bandwidth has to be maximised by the particular learning techniques. Details on the scenario and the control of BitTorrent in general are given in Chapter 7.3.3.

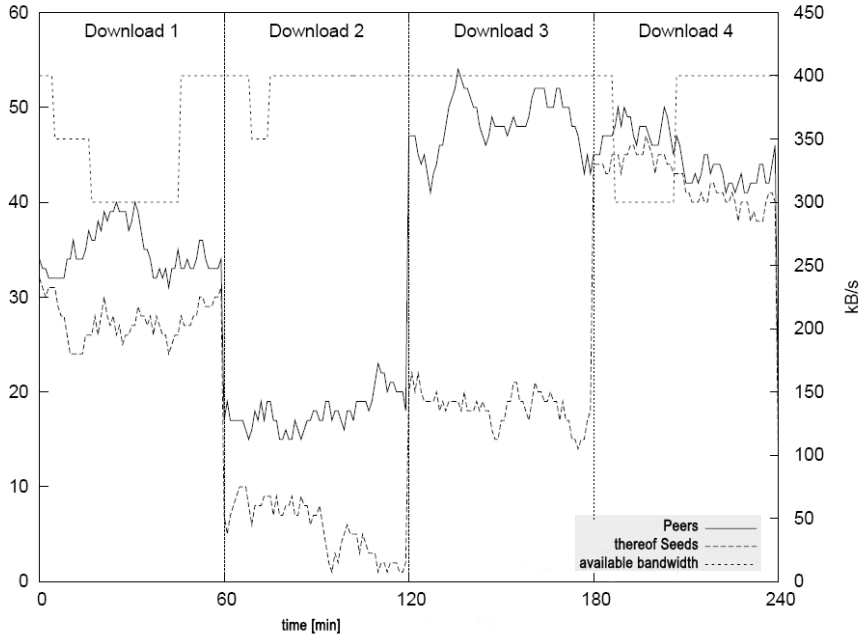
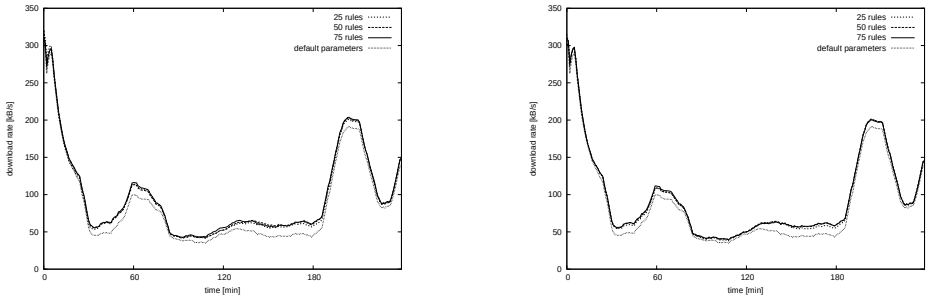


Figure 4.5: Test scenario for the on-line learning comparison: downloads and available bandwidth

The first part of the evaluation investigates the influence of available knowledge on the performance achieved by the techniques LCS and FCS. In order to provide the initial knowledge, 75 different situations occurring in the scenario have been chosen randomly and an off-line optimisation using the Layer 2 component has been performed. Afterwards, Layer 2 has been deactivated for the simulation. This knowledge has been provided at startup as available rules in the population of both classifier systems. In order to distinguish between different amounts of available knowledge, the simulation scenario has been processed with three different settings: 25, 50, and 75 available rules (the first two subsets are generated randomly from the last one). As illustrated by Figure 4.6, both techniques benefit from additional knowledge: with increasing population size, the bandwidth utilisation becomes more accurate. Figure 4.6(a) describes the achieved results for the modified LCS, Figure 4.6(b)

those of the FCS. The LCS resulted in an averaged performance of 92.8 $KByte/s$ with 25 rules available (50: 93.8 $KByte/s$; 75: 94.8 $KByte/s$), which is an increase of 10.6 % (50: 11.8 %; 75: 13.0 %) compared to the performance achieved by the protocol's standard configuration (89.3 $KByte/s$). In contrast, the FCS results in slightly worse results. The averaged performance has been increased to 91.5 $KByte/s$ with 25 rules available (50: 92.5 $KByte/s$; 75: 92.6 $KByte/s$), which is an increase of 9.0 % (50: 10.3 %; 75: 10.4 %).



(a) Performance of the LCS-controlled variant (b) Performance of the FCS-controlled variant

Figure 4.6: Comparison of LCS and FCS controlling a BitTorrent client

The second part of the evaluation deals with the question of how well the targeted bandwidth utilisation has been achieved. Therefore, Figure 4.7 depicts the averaged results for all three techniques (LCS, FCS, and ANN) distinguishing again between 25, 50, and 75 available rules. Layer 2 has been deactivated to analyse the pure performance of the on-line adaptation performed by the learning technique. As a comparison, the bandwidth utilisation using the protocol's standard configuration (the configuration is listed in Table 7.3) resulted in an averaged download rate of 89.3 $KByte/s$. The figure demonstrates the advantage of the LCS in comparison to the other two techniques. In all cases, the performance achieved by the LCS is the best one. In addition, a continuous increase in the performance can be observed for the rule-based techniques (LCS and FCS), while the ANN seems to run into the "overfitting" problem [144] – its performance does not necessarily increase with more rules (cf. values for 75 rules). Typically, the effort to achieve the results is considered besides the pure results, but this has been identical in the presented setting: preparation and training are done using the Layer 2 component.

A third aspect in the evaluation is the question whether the learning techniques choose inappropriate parameter settings or not. In this context, *inappropriate* is defined as decreasing the bandwidth utilisation compared to the performance achieved when using the protocol's standard configuration. Again, the LCS shows the best results, since such an inappropriate parameter set has only been chosen in 56 (55; 48) cases for 25 (50; 75) available rules. This means, providing only 75 rules leads to a better performance in about 80 % of the 240 cases. In comparison, the FCS selects an *inappropriate* parameter configuration

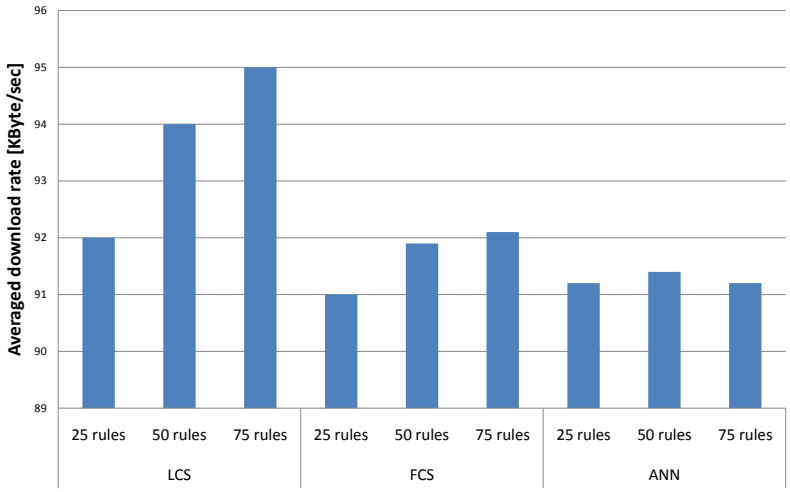


Figure 4.7: Comparison of LCS, FCS, and ANN with increasing knowledge

in 23 % and the ANN in 28 % of the 240 situations. Selecting inappropriate parameter settings in about 20 % of the situations sounds quite high, but this is caused by the limited knowledge base. Therefore, the last part of the evaluation considers the improvement of the selection quality during the operation period.

To analyse the quality of the selection process, the scenario has been processed in a re-occurring sequence. Within this setting, a continuous decrease in selecting inappropriate parameter settings has been observed for LCS and FCS with activated Layer 2, while the ANN keeps the current level due to the pretraining and lack of taking on-line feedback into account. Especially after training the populations of LCS and FCS using several iterative runs of the scenario, the quality of the selection process has been significantly increased and simultaneously the number of selected inappropriate parameter sets has been reduced to 0 (LCS) and 3 (FCS - about 1 %), respectively. The LCS selects always the best possible rule after 50 runs of the scenario leading to a nearly optimal bandwidth utilisation. The performance of the FCS is only slightly lower. In contrast, the performance of the ANN stays constantly at the level illustrated by Figure 4.7.

4.1.7 Summary: Automated Learning

The previous part of this thesis investigated the design choice for Layer 1 of the proposed framework: which machine learning technique provides the best results when covering the learning part of the controller? Therefore, the specific characteristics of the learning problem (especially in comparison to prominent artificial problems from literature) have been presented. Based on these characteristics, existing directions in machine learning have been discussed and candidate solutions have been identified. As a result of this analytic process,

rule-based learning techniques – in particular, Learning Classifier Systems (LCS) and Fuzzy Classifier Systems (FCS) – have been chosen for the usage within the framework. In order to fulfil all requirements, modified variants of both approaches have been developed taking especially the safety restrictions into account.

In order to identify the most promising technique, a comparison in an exemplary scenario has been performed taking further techniques into account, although they are not applicable due to e.g. safety restrictions. The scenario for the comparison is taken from the domain of dynamic network protocol control as explained in detail in Chapter 7. Although the presented results have been generated only for one scenario, they are meaningful since the control problem and the corresponding learning problem are assumed to be comparable (see Chapter 8.3).

As a result of the comparison, the modified LCS leads to the best observed performance. In addition, the safety restrictions are not violated and the initially formulated requirements (see Chapter 3.3.2) are fulfilled. Thus, the modified LCS will be used as on-line learning component of Layer 1.

4.2 Off-line Optimisation Component

The automated learning performed by the proposed framework consists of two parts: the *on-line improvement* of the selection strategy and an *off-line generation* of new actions for previously unknown situations – the *sandbox*. This sandbox is characterised by a similar design choice as previously discussed for the machine learning techniques. As depicted in Figure 3.2, a simulation tool is combined with an optimisation heuristic. The simulation part serves as evaluator and is application specific – in contrast, the optimisation heuristic is application-independent.

Since the framework provides a solution for controlling different types of SuOCs and each SuOC can be characterised by an (in principle) unique search space, a generalised description of search space and evaluation function are hardly possible. But, typically search spaces of real-world systems consist of a large number of dimensions leading to a complex *optimisation problem*. Considering the search space as arbitrary without any a-priori information, the *No-Free-Lunch-Theorem* [176] states that no better or worse search algorithms exists: no optimisation algorithm has significant advantages or disadvantages compared to any other algorithm, if they are applied to all possible search spaces and evaluation functions. Typically, all possible evaluation functions and search spaces do not reflect the particular problem to be optimised, no matter how broadly defined it is [177]. Thus, it does make sense to search for a preferably good technique for the considered class of problems.

Application problems from the real-world domain are characterised by a correlation between neighbouring points of the search space. More formally, two values $f(\overrightarrow{x})$ and

$f(\overrightarrow{x}^l)$ of the evaluation function for two nearby (in terms of the search space) elements $\overrightarrow{x}, \overrightarrow{x}^l$ are seldom deviating largely [178, 179]. Such a correlation is also assumed for the search spaces of the particular control problem. Based on this assumption, a comparison of search heuristics can be performed looking for the best results in an exemplary setup for Layer 2. The remainder of this section investigates which optimisation heuristic promises the best results when applied to the sandbox's optimisation task.

4.2.1 Term Definition: Optimisation Problem

The optimisation problem to be solved by the Layer 2 component is based on the search space S and a given evaluation function $f(\overrightarrow{x})$ and can be defined as follows:[1]

$$f(\overrightarrow{x}) \rightarrow max;\ \overrightarrow{x} = (x_1, x_2, \ldots, x_D) \in S \subseteq \mathbb{R}^D; x_{min,i} \leq x_i \leq x_{max,i}, i \in [1, \ldots, D] \quad (4.4)$$

with $\overrightarrow{x}$ describing the parameter vector for the SuOC, D denoting the dimensions of the parameter vector, and $x_{min,i}$, $x_{max,i}$ being the minimum and maximum boundaries for the parameter x_i. The evaluation function $f(\overrightarrow{x})$ quantifies the quality of a point $\overrightarrow{x}$ within the search space S. In addition, the general optimisation problem is also defined for an aggregated evaluation function consisting of several single evaluation functions – in the context of this thesis, only single evaluation functions $f(\overrightarrow{x})$ are considered.

The **global optimum** $\overrightarrow{x}^*$ (the "best" solution) of the optimisation problem is defined as:

$$\forall\ \overrightarrow{x}\ \in\ S:\ f(\overrightarrow{x})\ \leq\ f(\overrightarrow{x}^*) \quad (4.5)$$

This means, there exists no other parameter configuration of the SuOC in S leading to a better function value of f than $\overrightarrow{x}^*$. In contrast, a **local optimum** $\overrightarrow{x}^l$ defines an optimum in a restricted environment:

$$\exists \epsilon > 0 : \overrightarrow{x}\ \in\ S:\ p(\overrightarrow{x}, \overrightarrow{x}^l) < \epsilon\ \Rightarrow\ f(\overrightarrow{x})\ <\ f(\overrightarrow{x}^l) \quad (4.6)$$

with p a distance measurement within the search space S. Hence, a parameter set $\overrightarrow{x}^l$ is the best solution within an ϵ environment. As summary, the target of Layer 2 is to find the global optimum $\overrightarrow{x}^*$ for a given setting and a given function f by avoiding local optima.

[1] The minimisation can be solved equally by maximising $-f(\overrightarrow{x})$

4.2.2 Overview: Optimisation Heuristics

Optimisation is a well-known research domain attracting researchers mainly from mathematics and computer sciences for decades. Thus, several different concepts have been presented and a full listing of all of them is unrewarding. In contrast, a classification of existing techniques provides a short overview of different directions in research. According to *Weise* [178], the two main categories of *deterministic* and *probabilistic* approaches can be distinguished. **Deterministic algorithms** are used, if a close connection between solution and problem exists or a-priori information about the search space is available. In this case, the search space can be partitioned according to schemes like *Divide-and-Conquer* [180] and the optimisation problem can be solved using deterministic algorithms like *Branch-and-Bound* [180]. In addition, research has developed deterministic methods to decide whether a global optimum has been found or not [181]. The drawbacks of these algorithms are visible whenever the search space is too vast, when it is characterised by many dimensions, or when there are no (or only loose) connections between the different points within the search space: the algorithms do not scale anymore.

In contrast, **probabilistic algorithms** use (at least at one position) randomised values – consequently they are heuristics that might not always lead to the best solution. But their advantage is the handling of the drawbacks named before. In addition, they are often more efficient and have a less complex implementation, while the solutions found are typically only slightly deviating from the optimum. This is accompanied by the advantage of a significantly faster delivery of the result. In most cases, probabilistic algorithms contain *heuristics* or *meta-heuristics* providing the possibility to approximate solutions without mathematical proof of convergence [179]. These heuristics take already derived knowledge into account to decide about the next candidate to be tested. Furthermore, a *meta-heuristic* is a technique to solve abstract problem classes more efficiently. Therefore, abstract heuristics are combined with an evaluation function without allowing a deeper insight into the particular structure of the evaluation functions [176].

The class of probabilistic algorithms can be further partitioned according to the way the heuristic works. Besides several less-famous concepts, two broad categories can be distinguished: *population-* and *trajectory*-based approaches [126]. **Population-based** approaches use a set of different self-motivated entities looking for the best solution. Due to the distribution of the search problem to several entities, the approach is more robust than individual search and typically easier to parallelise. The knowledge stored in the particular individuals can be used to direct the search and make use of cooperative behaviour.

Evolutionary Algorithms (EA) [116] mimic the evolution of biological creatures using the three basic concepts *selection*, *crossover*, and *mutation*. Based on a randomly created population, the following loop of steps is continuously carried out by the EA until a stop criterion is reached: 1) the EA determines the values of the objective function for each solution candidate, (2) the EA selects the fittest individuals from the population according to

their fitness values for reproduction of new offspring solutions using crossover and mutation, and (3) the EA checks whether a termination criterion is met or not (e.g. a maximum number of function evaluations) and either returns the best solution found so far or repeats the steps from 1 to 3. Based on this working principle, different directions of EAs can be distinguished with *GAs* and *Evolution Strategies* the most famous ones [182]. EAs have been successfully applied to a wide range of application domains from function optimisation [183] to image processing [184].

Differential Evolution (DE) is another population-based optimisation algorithm, initially proposed by *Storn and Price* [185]. The approach is founded on a different concept to generate new parameter vectors. Instead of genetic-based reproduction, new parameter vectors are generated by adding a weighted difference vector between two population members to the parameter vector of a third one.

Let $x \in \mathbb{R}^n$ be one individual parameter vector describing a possible solution in the population. Then, the DE algorithm can be described as follows: 1) all candidates x_i are initialised at random positions of the search space, 2) the following loop is repeated for each individual x_i in the population until a termination criterion is met:

- Select randomly three distinct agents a, b, and c from the population
- Choose a random index $R \in 1, \ldots, n$, with n equals the number of dimensions
- Compute x_i's new position $y = [y_1, \ldots, y_n]$ by iterating over each $i \in 1, \ldots, n$:
  - Choose a random number r_i uniformly distributed in the range (0,1)
  - If $(i = R)$ or $(r_i < CR)$, set $y_i = a_i + F(b_i - c_i)$; otherwise set $y_i = x_i$
  - If $(f(y) < f(x))$, set x = y in the population (replace)
- Return the best solution found so far

Within the previous algorithm, $F \in [0, 2]$ denotes the *differential weight* and $CR \in [0, 1]$ the *crossover probability*, both parameters and the *population size* ($NP > 3$) are configurable in advance. DE has been successfully applied to several real-world applications [186].

A third *population*-based optimisation algorithm has been developed by *Kennedy and Eberhart*: **Particle Swarm Optimisation** (PSO) [187]. In contrast to the evolution-driven approaches before, this algorithm is inspired by the *flocking of birds*, where individual birds spread in the environment to look for food and move around independently from the other group members. The environment of the birds corresponds to the search space of the optimisation heuristic, while each bird is represented as an individual. Each individual (the *particle*) keeps track of the best solution it already discovered and the best global solution found so far. In addition, it is propelled towards these positions at each optimisation step. To choose the next position within the search space, each particle has to determine a *velocity vector* according to the following formula:

$$V_{k+1}^i = \omega * V_k^i + \varphi_1 * U_1[0,1) * (P_{id} - X_{id}) + \varphi_2 * U_2[0,1) * (P_{gd} - X_{id}) \tag{4.7}$$

where V_k^i represents the current *velocity* of the particle, ω stands for the *inertia* weight, X_{id} for the current *position* of the particle, P_{id} for the *personal best position* of the particle, and P_{gp} for the *best global position* found so far. φ_1 and φ_2 are the *acceleration coefficients*, while $U_1[0,1)$ and $U_2[0,1)$ are uniformly distributed random numbers generated between 0 and 1 (excluding 0). PSO is a well-known technique and has already been applied to different real-world applications [188].

Harmony Search (HS) [189] is another *population*-based search heuristic, which is inspired by jazz musicians trying to find a harmonic consonance of their instruments [190]. The approach consists of five iterative steps: 1) generate random vectors $(x^1, \ldots, x^{hms})$ with *hms* defining the size of the *Harmony Memory* (the population size), store them in the Harmony Memory (HM):

$$HM = \begin{bmatrix} x_1^1 & \ldots & x_n^1 & | & f(x^1) \\ \vdots & \ddots & \vdots & | & \vdots \\ x_1^{hms} & \ldots & x_n^{hms} & | & f(x^{hms}) \end{bmatrix}$$

In step 2, a new vector $\overrightarrow{x}^l$ is created by choosing each contained x_i as follows. With the given probability *hmcr* (harmony memory considering rate), a randomly chosen entry from column i of HM is selected – otherwise $(1 - hmcr)$, a random value within the allowed range is picked. Step 3 performs randomised mutations with a given probability and step 4 checks whether the performance $f(\overrightarrow{x}^l)$ of the new vector $\overrightarrow{x}^l$ is better than the worst one contained in HM – in this case, the worst one is exchanged by $\overrightarrow{x}^l$ (otherwise $\overrightarrow{x}^l$ is neglected). Afterwards, step 5 defines a loop by repeating steps 2 to 4 until a stop criterion is reached. HS has been successfully used e.g. in therapeutic medical physics to optimise radiotherapy of cancer tumours [191].

The most prominent representative of *trajectory*-based optimisation algorithms is **Simulated Annealing** (SA) [192] – also known as *Monte Carlo annealing* or *probabilistic hill climbing*. The approach is based on an analogy taken from thermodynamics – SA uses a temperature-based approach to escape local optima. In general, SA accepts new solutions in two cases: 1) if they are better as the currently known best one or 2) if a bad solution fulfils the following condition:

$$random[0,1) \; < \; e^{\frac{eval(v_n) - eval(v_c)}{T}} \tag{4.8}$$

where v_n represents the new solution, v_c represents the current solution and T describes the temperature. *Eval* is the evaluation function. T is continuously decreased over time leading to a corresponding decrease in the probability to accept "bad" solutions. Due to the decreasing temperature and the correspondingly decreasing probability to accept new

solutions, SA converges according to the chosen cooling rate. At the end of the optimisation, the temperature may be so low that the final stages of SA merely resemble an ordinary hill-climbing algorithm. The concept has been successfully applied to different scenario domains from combinatorial optimisation to machine learning [193].

Besides these most prominent algorithms, several further approaches can be found. Especially nature-inspired techniques have generated attraction within the last years. But most of them can be reduced to *population*-based or *trajectory*-based approaches. In addition, most of them have been developed or adapted to solve special problems. Thus, the focus of this thesis is set on comparing the standard approaches from literature.

4.2.3 Comparison of Optimisation Heuristics

In contrast to machine learning techniques, an exclusion of categories or single techniques based on analytic considerations is hardly possible for optimisation algorithms. Only deterministic approaches can be excluded due to to their limitations considering vast search spaces. Thus, the remainder of this section presents an empiric comparison of the most famous techniques in a representative setting for the developed framework. Therefore, the presented investigation is based on the work done by *Domdey* in [194] and re-uses the scenario.

Similar to the previous part of this chapter, the second part of the learning problem (the Layer 2-situated optimisation problem) is assumed to be comparable for all systems applicable to the developed framework. In particular, the relation between situation and best interrelating action corresponds to the functional model as introduced in Chapter 8.3. Thereby, it is assumed that all investigated applications are of the same type. Thus, one of them can be analysed as representative for the complete group. As indication for this assumption to be correct, the reader is referred to *Reeves* [195] and *Kauffman* [196]. *Kauffman* investigated the characteristics of real-world fitness landscapes in general and introduced the *N K-landscapes* [196]. Based on these considerations, *Reeves* states that fitness landscapes of real-world applications are comparable and quite similar in most cases.

Again, the environment of the Organic Network Control (ONC) system as discussed in Chapter 7 is used as foundation for the comparison. In mobile ad-hoc networks, the optimal parameter configuration of the protocol depends highly on the distribution of other nodes in the direct neighbourhood. Thus, Layer 2 might receive a snap-shot of nodes in the direct neighbourhood (as observed by the observer of Layer 1) to find the best parameter configuration for this situation. Figure 9.2(b) depicts the setup for one scenario (Scenario 4). The node to be optimised is situated in the centre of the sector-model, each point represents a neighbour. The arrow describes the moving direction of the node. The evaluation is based on simulations performed in the standard network simulation tool *NS-2* [130], with *NS-2* serving as operating part of the evaluation function (the processing of the particular

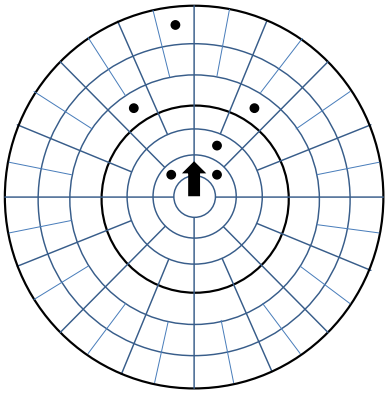


Figure 4.8: Exemplary optimisation scenario (scenario 4)

search techniques makes up only a small fraction of largely less than 5% of the evaluation duration, the rest is spent on NS-2 runs). Details on the scenario and the corresponding control of a mobile ad-hoc network protocol are given in Chapter 7.3.1. In order to determine appropriate test scenarios for the comparison, a set of different situations has been chosen. Appendix B describes all scenarios in detail and lists the achieved results of all compared techniques.

The comparison includes the previously introduced techniques EA, DE, PSO, HS, and SA. Therefore, the algorithms have been implemented according to the basic design presented by the particular authors. All of these techniques can be customised to specific optimisation problems by choosing a set of configuration parameters. Since the target of the comparison is to find the best technique for the analysed scenario, the best parameter configuration for each algorithm has to be determined in a first step. Therefore, a parameter study has been performed choosing that setup providing the best results for all considered scenarios. Thereby, a similarity of the scenarios has been observed – in most cases, one parameter configuration prevailed against all other tested configurations in each scenario. Thus, the investigated scenarios seem to provide quite similar challenges to the search techniques. Another supporting aspect for the setup using the configurations found for the algorithms is that they are in conformity with those recommended in literature. The configurations of the particular techniques are listed as follows.

The EA is realised as a GA, using a *population size* of 10. In each generation, five new individuals are generated as offspring using *one-point cross-over*, *ranking list selection*, and a *continuous state model* with *elitist strategy*. New individuals are mutated according to [182] based on a Gaussian-distribution model. Thereby, a standard deviation of 10% of the particular dimension is considered. The DE algorithm is realised in the *DE/rand/1/bin* variant using a *population size* of 20, a *scaling factor* of 0.9, and a *cross-over* constant of 0.2. These values correspond to the configuration as described in [197]. For the HS algorithm, the size of the *harmony memory* has been set to 10. The *harmony memory consideration*

rate is 0.85 and the *pitch adjusting rate* is 0.45 with 10 % bandwidth in each dimension. This setup corresponds to the configuration as given in [190].

The PSO technique has been initialised with 20 particles. Their *inertia* has been defined with 0.73, while the constant c_1 has been set to 2.8 and c_2 to 1.3. Furthermore, the *maximum velocity* $\overrightarrow{v}_{max}$ has been defined as half of the spatial extent of each dimension. The topology of the neighbourhood is defined as a *ring-based topology* with an asynchronous update of the particles – these configurations correspond mainly to those described in [198, 179]. For all population-based search techniques, a preferably small population size has been searched in order to allow for finding first results quickly. In general, large population sizes do not provide a higher benefit, since the calls of the evaluation function restricts the search [182].

The SA algorithm starts with an initial temperature of 20 *degrees*, with a cooling taking place each five calls of the evaluation function. The configuration parameter α has been set to 2 and the parameter K to 500. The selection of a point in the neighbourhood is done using a *direction cosine*, the *scaling factor* corresponds to 10% of the spatial extend of each dimension. These configurations can also be found in [199, 178].

The Layer 2 component has to provide solutions as good as possible and has to find them as fast as possible. Thus, the comparison is based on evaluating these aspects. The **success rate** (SR) defines the percental fraction of successful optimisations. An optimisation is called successful, if the optimal or an approximately optimal solution (solutions approximating the optimum to a certain degree) has been found. Since the optimal setting is typically unknown for protocol configurations in given situations, the performance of the standard protocol configuration is used as reference. In addition, the **averaged performance** (AP) is a measurement to describe the averaged success of the algorithm. It is calculated as average of the best candidates' performances in all runs. Both measurements (AP and SR) quantify the success of the algorithm; in addition, the *speed* of its success is needed. Therefore, the **number of fitness function calls** (NC) is taken into account, which counts how often the NS-2 simulator has been used until the best candidate (the final result) has been found.

As a further metric for the success of the techniques, a random-based search strategy (*Rand*) has been implemented. Here, a randomly chosen parameter configuration is tested using the NS-2-based evaluation function. Only the best solution found so far is stored – which is returned when queried. The different comparison measurements (SR, AP, NC) have been defined according to those proposed in literature when comparing optimisation techniques [182]. All results presented in the remainder of this section have been determined as averages of 100 different runs. The comparison has been performed on a 2.66 GHz dual-core machine with 4 *GByte* RAM and openSUSE 10.3 as operating system.

Criterion 1: Averaged Performance

The first aspect covered by the evaluation is the *averaged performance* achieved by the particular techniques. In order to quantify the performance in relation to the number

of calls of the evaluation function, two different measurement series have been analysed: the achieved performance after 83 calls of the fitness function and after 500 calls. Both values have been determined by another parameter study. The former value (83 calls of the evaluation function) has been chosen since it describes a lower boundary for the optimisation technique. From this point on, at least a subset of the heuristics results in stable good solutions compared to the protocol's standard configuration. The Layer 2 component of the framework has to work under real-world conditions, which makes a fast reaction necessary – thus, one aspect is to quickly find good solutions. In contrast, the latter value (500 calls of the evaluation function) describes a trade-off for an upper boundary. Although not always the best solution is found, the majority of the heuristics has converged to approximately optimal solutions.

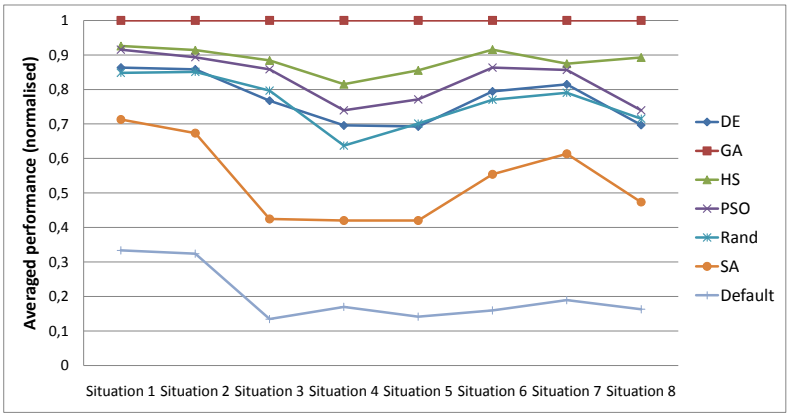


Figure 4.9: Averaged performance after 83 calls of the evaluation function (higher values are better)

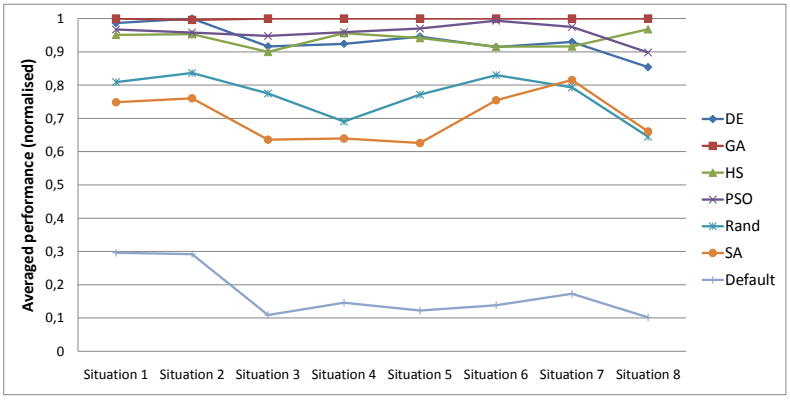


Figure 4.10: Averaged performance after 500 calls of the evaluation function (higher values are better)

Figure 4.9 describes the results after 83 calls of the evaluation function. The y-axis depicts the normalised achieved performance of the candidate techniques, while the x-axis lists the different investigated scenarios. The performance has been normalised as follows. The achieved fitness has been calculated for each scenario in relation to the best achieved value as reference (100 %) – thus, the different scenarios become comparable. Besides the five techniques to be compared (DE, GA, HS, PSO, and SA), the figures name two further candidates: *Rand* and *Default*. The former one is the previously mentioned random search strategy, while the latter one represents the performance achieved by using the protocol's standard configuration (the static setup as proposed by the authors of the protocol).

The results depicted in the figure show that the considered GA finds the best solutions in all scenarios, followed by HS and PSO. Surprisingly, the random technique leads to better results than SA, while (as expected) the protocol's standard configuration has been outperformed by all techniques. Similar results can be observed when increasing the number of calls of the evaluation function to 500. Figure 4.10 depicts the results showing again that the GA finds the best solutions. In contrast to the previous figure, the range of the best solutions is significantly smaller: DE, HS, and PSO find nearly the same (best) parameter configurations. Only SA declines from the rest as it results in solutions comparable to the randomised strategy. The standard deviation observed for all 100 runs of the optimisation per candidate technique is very small, a significant deviation is observed in scenario 8 only. A small standard deviation means that all runs lead to comparable results. Scenario 8 differs, since it seems to be the most complex optimisation task in this context. For instance, the scenario contains a higher number of nodes – details are depicted in Appendix B.

Criterion 2: Calls of the Evaluation Function

The second criterion describes how many calls of the evaluation function are needed until a *successful* solution has been found. Even in literature, there is no commonly agreed definition for *successful* [182]. Intuitively, one could define *successful* as better than the reference value (e.g. the performance achieved by using the protocol's standard configuration), but this is not useful since each optimisation technique results in relatively better solutions only after a few calls of the evaluation function. The consideration of an additional factor ϵ to increase the basis for comparison (default performance + ϵ) has proven to be not applicable due to the variability of the different scenarios. In some scenarios only a few percent increase is found compared to the reference value, while in others it has been largely outperformed. Hence, a relative measurement has been chosen quantifying the performance according to the achieved results in the particular scenario. The simulation is stopped for each technique if it finds the first solution that is only ϵ % worse than the best averaged solution found for this technique in 10 optimisation runs with 500 calls of the evaluation function ($\epsilon = 10$ %). Thus, the metric describes how fast the technique converges in relation to the best solution it can find after 500 calls of the evaluation function.

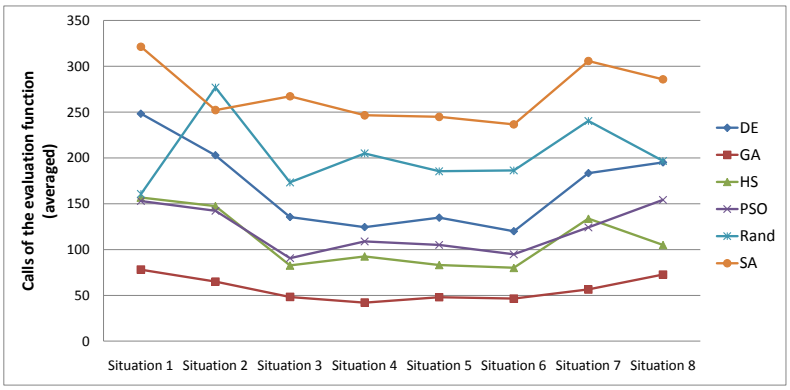


Figure 4.11: Averaged calls of the evaluation function to find successful candidates (lower values are better)

Figure 4.11 illustrates the results. Compared to the previous diagrams, the design is adjusted concerning the y-axis. It depicts the averaged number of calls of the evaluation function until a successful solution has been found. The relative measurement for successful candidates is reflected by the results of the previous evaluations. The GA requires the smallest number of calls, while HS, DE, and PSO result in approximately the same number of calls. Again, SA performs at a comparative level to the randomised strategy.

Criterion 3: Success Rate

Finally, the success rate of the optimisation techniques is considered describing which fraction of the 100 optimisation runs has been successful. Again, *successful* is defined according to the previous evaluation criterion. Hence, it is a relative measurement for the set of investigated techniques. Figure 4.12 depicts the results after 83 calls of the evaluation function and Figure 4.13 after 500 calls – in both cases the y-axis describes the success rate. Both figures illustrate again that the EA is the most promising technique. Only in scenario 2 (500 calls), the EA did not lead to the best success rate. Here, DE and HS showed a slightly better performance.

4.2.4 Summary: Optimisation Heuristics

The previous part of this thesis investigated the design choice for Layer 2 of the proposed framework – which optimisation heuristic provides the best results when covering the optimisation part of the sandbox-learning component? Since this selection cannot be solved analytically like for the on-line learning component in Chapter 4.1, an extensive comparison in an exemplary scenario has been performed. This scenario is taken from the domain of dynamic network protocol control as explained in detail in Chapter 7. Although the pre-

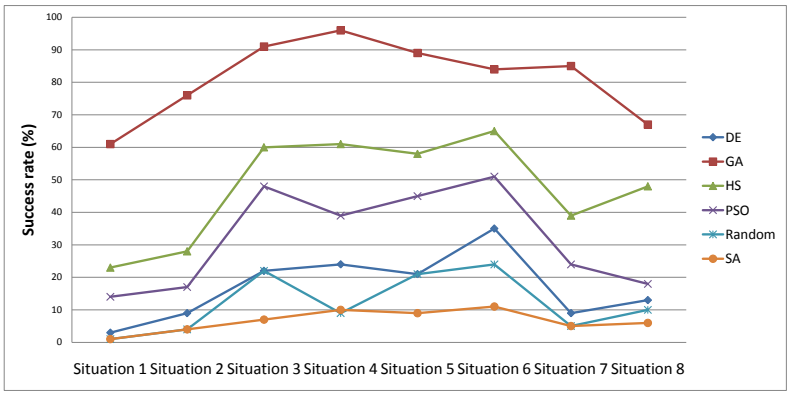


Figure 4.12: Success rate after 83 calls of the evaluation function (higher values are better)

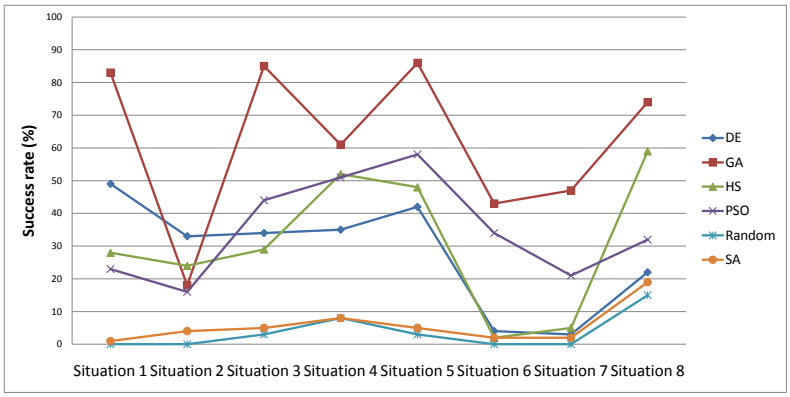


Figure 4.13: Success rate after 500 calls of the evaluation function (higher values are better)

sented results have been generated only for one scenario, they should be meaningful: they belong to the same type of control problems and consequently the corresponding learning and optimisation problems are assumed to be comparable.

Considering the results as presented in Chapter 4.2.3, the usage of *EAs* (in particular, a GA) seems to lead to the most promising results. In nearly all investigated scenarios, this technique has found the best results. In addition, it needed the lowest amount of evaluation function calls. Both figures describe the special requirements for optimisation techniques applied to the sandbox. Typically, fast results are needed to extend the behavioural repertoire of Layer 1 and these new behaviours have to be as good as possible. Therefore, the previously described analysis of the achieved results is based on two different optimisation durations: a short one (lower quality results, but very fast) and a longer one (good results, about 6 times slower).

For the configuration of the optimisation heuristic within the framework, the question arises of how many calls of the fitness function should be allowed. Here, a trade-off between both aspects seems to be most promising. Initially at start-up, the system is typically confronted with a large set of previously unanticipated situations. Thus, a fast delivery of new behaviours by Layer 2 is needed (supporting a lower number of fitness function calls). In contrast, the rate at which new situations appear is significantly decreased after a certain period of operation – here, the time horizon of the optimisation task can be extended to allow for solutions with an ever higher quality. In addition, former behaviours can be reviewed and probably improved, which allows for a continuous self-improvement during the complete operation period of the system. As a summary, the GA will be applied to Layer 2 of the framework.

4.3 Design Recommendation

This chapter discussed design choices for two major components of the architecture: the *on-line learning* and the *rule-generation* component. Initially, the special characteristics of the learning problem of Layer 1 have been discussed, followed by determining promising candidate techniques from the set of existing machine learning techniques. From these, *LCS* (in particular the XCS variant by *Wilson* [159]) and FCS have been identified as most promising approaches. Based on these concepts, adapted variants have been developed matching the requirements of restricted on-line learning in the framework. These adapted variants have been compared in an exemplary setting to further ML techniques resulting in different performances of all candidates. As a result, the LCS variant has been proven as most promising approach and thus will be used in the remainder of this thesis.

In addition, the rule-generation component of Layer 2 (the “sandbox”) is composed of a simulation tool and an optimisation heuristic. Thus, the second part of this chapter focused on choosing the most promising optimisation heuristic to be used within the framework.

Therefore, the optimisation problem has been defined, followed by a short overview of existing techniques in literature. Since an analytic classification as done for the on-line learning part is not feasible, the selection process is based on an exemplary comparison only. As a result of this comparison, the developed variant of an EA leads to the most promising results and will therefore be used in the framework.

Chapter 5

Structure of the Evaluation

The previous chapter introduced the design of the system and investigated which particular techniques should be used to cover the tasks defined by the framework. The following three chapters address the evaluation of the previously introduced framework. An analysis of the framework's performance requires an underlying application scenario. Therefore, the following chapters introduce two application scenarios in detail and present an overview of two further application scenarios. Based on the former two examples, the question of how the system's performance is influenced by applying the framework to the parametrisable system will be answered. Therefore, domain-specific metrics of the particular area will be used to analyse the behaviour.

The first application scenario is the Organic Traffic Control (OTC) system, which is designed to control traffic lights at urban intersections without the need of network-wide knowledge (see Chapter 6). Afterwards, the Organic Network Control (ONC) system (see Chapter 7) is introduced, which adapts network protocol parameter configurations dynamically and in response to changing environmental conditions. Besides the impact of domain-specific metrics, a second criterion is the effort needed to achieve the adaptive behaviour. Therefore, the number of rule-generations, the performed parameter adaptations, and the size of Layer 1's rule base will be analysed.

Furthermore, the question arises for which parametrisable systems the framework is applicable – and in which cases problems might occur. This question is investigated following two different directions. On the one hand, two further application scenarios are introduced to generalise the approach (an Organic Production System and an error-prediction system for mainframes). Afterwards, the framework is evaluated for an abstracted mathematical problem. On the other hand, aspects of the OTC and ONC scenarios are used. In ONC, wireless sensor nodes are investigated as one possibility to apply the framework to systems

with hard resource constraints. Typically, applications running on a sensor node are limited in terms of available computation, storage, and communication resources. Within this scenario, the minimally required resources for applying the framework are analysed.

One basic concept within the design of the framework is to restrict the exploration part of the learning technique by introducing a simulation-based sandbox component. In order to figure out the limitations of this approach, the ONC system is applied to the control of the Peer-to-Peer protocol **BitTorrent**. A BitTorrent client has only limited information about its neighbourhood, the network's status, the network's design (in terms of available links and bandwidths), and the status of the neighbours. In addition, the *neighbourhood* is not locally restricted like in OTC. Hence, it consists of significantly more neighbours and unknown interdependencies. Consequently, it is expected that the simulation-based approach will show its drawbacks for systems with such inadequate neighbourhoods.

Another important capability of the framework is to deal with disturbances and noise. Hence, already the first scenario (OTC in Chapter 6) distinguishes between a "normal" and a "disturbed" scenario. This classification is also used in the second application example (ONC in Chapter 7). Thereby, the exemplary customisation of ONC for wireless sensor nodes has been chosen to investigate whether there are differences in the performance with increasing dynamics or not. In addition, the impact of noise is covered since OTC works on the basis of data obtained from (simulated) sensors. Finally, Chapter 8.3 will analyse the framework's behaviour at a more abstract level by applying it to a generalised mathematical model. Within this model, both aspects – noise and disturbances – can be analysed in an isolated manner detached from other influences of the learning behaviour.

Chapter 6

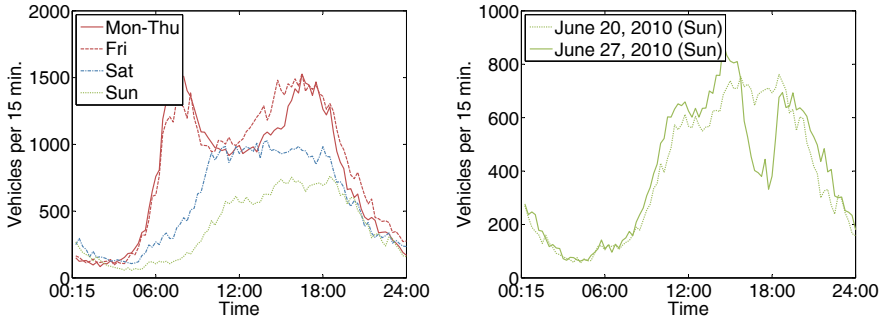
Organic Traffic Control

Due to the dynamic characteristics of urban transportation (both, individual traffic and public transport), the control and management of traffic control systems for urban road networks is a complex phenomenon. In particular, many varying and possibly conflicting objectives cause this complexity – which has attracted research from different domains of science (including civil engineering, physics, and informatics) to investigate solutions for and characteristics of the underlying problem. Many participants (e.g. car drivers, cyclists, or bus drivers) are constantly interacting in parallel leading to difficulties in predicting the future state of the complete transportation system. In combination with the interdependencies resulting from the impact of one's decision on the other's behaviour, approaches with standard optimisation techniques are not promising since they tend to give obsolete solutions. Usually, the problem of which control strategy is *best* changes before it can be optimised.

Traditional approaches proposed to cover control tasks in urban traffic systems (like *Scoots* [200] and *Scats* [201]) have focused on a centralised system structure. Consequently, they are characterised by natural limitations in terms of scalability and response times since the central computing power necessary for these tasks grows significantly with the system size. In addition, the amount of data to be collected in real-time and the transmission to a centralised location are further limitations. Especially, if the target is to find a global optimum for the behaviour of all contained control units within a traffic network, a centralised approach becomes infeasible with growing size of the network [202, 203]. Even networks of small cities are often too complex to determine a matching solution in real-time. In contrast, decentralised solutions promise faster reaction times due to a locally restricted sensor horizon and consequently a less complex optimisation problem. Research has already generated considerable output for locally organised traffic control, but so far only small networks with

reduced functionality or proprietary solutions have been investigated [204, 205, 206]. Therefore, the optimisation of traffic control strategies has become part of the focus of initiatives considering decentralised approaches (like OC [207]) and serves as first application scenario in the context of this thesis.

One example for the limitations of current traffic control systems and their installations is given by Figure 6.1. Reoccurring patterns in traffic (as depicted in Figure 6.1(a)) can already be handled by existing urban traffic control systems, since they are known at design time and can be covered by responsible traffic engineers. Therefore, one possible concept is a time-dependent switching of signal plans. In contrast, Figure 6.1(b) picks up an example from reality [208] and illustrates the drawbacks of preplanned strategies and schedules. The figure compares the traffic demands of an arterial road at Karlsruhe, Germany, for two subsequent Sundays. June 20, 2010 has been a regular Sunday and the corresponding traffic profile has been handled according to the preplanned schedule in a satisfying manner. The irregularity occurred one week later in the afternoon of June 27, 2010. Germany played England in the round of last sixteen of FIFA's World Cup 2010. The game took place at Bloemfontein, South Africa, but also affected the traffic in Karlsruhe. Until noon, the observed traffic is similar to the previous week, which serves as standard profile. Afterwards, a dramatic change is visible: an unanticipated peak before and a dramatic drop of traffic after begin of the match. Such unanticipated behaviours cannot be handled with time-dependent solutions and consequently require more context-aware and autonomous solutions.



(a) Regular demand (b) 2010 FIFA World Cup

Figure 6.1: Traffic demand of an arterial road at Karlsruhe, Germany [208]

As a consequence of these considerations, the *Organic Traffic Control* (OTC)[1] System has been introduced as one approach for such a context-aware traffic control system. Based on the general design and the corresponding framework of this thesis, the system adapts the control strategy of traffic lights to changes in the observed traffic flows at the intersection. The initial concept of OTC has been presented in [163] and was continuously extended

[1]The OTC system has been developed within the DFG project "Organic Traffic Control" of the Priority Programme 1183 "Organic Computing". The project is a cooperation between Leibniz Universität Hannover, Institute for Systems Engineering and the Karlsruhe Institute of Technology, Institute AIFB. Research on OTC has been done cooperatively with *Fabian Rochner* (Hannover) and *Holger Prothmann* (Karlsruhe).

[164]. An overview of the concept in the context of autonomously and locally organised traffic control at urban intersections using OTC is presented in [2, 208].

Within this chapter, the OTC system is discussed with a special focus on the architecture and its application to the domain of traffic. Initially, the traffic control problem to be solved by OTC is introduced. Afterwards, related work considering traffic control from different points of view is discussed: *centralised* systems and *locally organised* approaches. Since the framework as presented in Chapter 3.3 requires a customisation to the specific application domain, Section 6.3 describes the necessary adaptations. In addition, collaborative extensions for the Layer 3 component are presented and analysed (see Section 6.4). Based on the resulting system, an experimental evaluation has been performed, which is presented in Section 6.5. Finally, the chapter concludes with a summary of the achieved results.

6.1 Problem Description

Considering existing solutions in urban traffic control, two main types of control strategies for intersections can be distinguished: *fixed-time controllers* (FTC) and *traffic-responsive control systems* (following e.g. the American *NEMA* standard [132] or the Swiss *VS-Plus* standard [209]). Both approaches rely on a cycle-based processing of control logic. The traffic lights of the turning movements are grouped into *signal phases*, which obtain the right of way (they are switched to show "green") in a reoccurring sequence. Additionally, *interphases* might be applied to afford the clearing of the intersection's conflicting area before other streams can affect this area. Thus, the *cycle time* is defined as the sum of all phases and interphases to be switched before the iterative process starts over. The difference between both approaches named before can be found by the durations of phases and their sequences – they are fixed for FTCs. The result is a constant cycle time. In contrast, traffic-responsive controllers can vary the phase durations and sometimes even the phase sequences. For instance, the American NEMA standard [132] defines the decision to switch to the next consecutive phase based on the queue length of currently detected waiting cars and based on gaps in the approaching traffic.

The OTC system is designed to autonomously control each intersection of the network without the need of a centralised element covering network-wide optimisations: each intersection controller is responsible for a limited area only. Thus, the control system for the particular intersection can learn from its actions and optimise its behaviour more effectively than solutions with a broader focus – especially centralised systems. Hence, the task of the OTC controller is to choose appropriate lengths for the durations of all phases contained by the local intersection's signal plan and consequently for the resulting cycle time. This selection is done according to the observed traffic situation at the particular intersection. The control of an intersection in order to minimise waiting times has been proven to be an NP-complete problem [210]. Since FTCs account for the major part of the installations in

Germany and because of a better availability of real-world data for comparison purposes, this thesis focuses on FTCs. However, the concept itself is also applicable to traffic-responsive installations.

Besides the pure local optimisation of the control strategy, decentralised collaborative tasks are part of the problem. Typically, control strategies of urban traffic networks are coordinated along main routes and during peak periods (main traffic hours in the morning and evening, see Figure 6.1(a)) forming *Progressive Signal Systems* (PSS, also called *green waves*). This coordination can have a beneficial effect on the traffic flows and the average number of stops per car travelling through the network. Therefore, Layer 3 of the architecture contains collaboration mechanisms being capable of e.g. establishing PSSs. Adapting the intersection's signalling and establishing a traffic-dependent coordination is a pure reactive task according to observations in the current traffic patterns. In order to develop a more integrated traffic management system, *route guidance* and *driver information* are necessary parts to achieve an active management. Therefore, OTC contains further Layer 3 mechanisms for these purposes.

Concluding the aspects named before, the requirements of the underlying problem to be solved by OTC have a high degree of similarity to those initially defined for the general framework. Considering e.g. the characteristics discussed by *Diakaki* for the *TUC system* [211], the demand of such a solution can already be found in literature:

- The system provides an efficient solution to traffic control in terms of fast reaction times and low additional computational needs.
- The system is robust with respect to possible measurement inaccuracies and disturbances.
- The system is reliable in case of hardware failures (detectors, communication links, etc.).
- The system can be easily deployed to a new installation without high adaptation effort.
- The system's behaviour is traceable – an engineer has the possibility to understand with reasonable effort what the system does in a given situation.
- The system is based on limited measurement requirements – ideally, it works based on current installations.
- The system works with low computational effort and communication requirements.

Thus, OTC's goal is the locally-organised control of traffic lights at urban intersections in combination with collaborative elements to achieve a network-wide optimisation of traffic flows. The system is implemented as a distributed solution and acts based on locally observed knowledge and measurements only. Therefore, it provides an ideal test case for the customisation of the general framework. The listed characteristics are provided by the OTC system due to decentralised operation and the customisation of the proposed framework.

Since traffic control is an own research domain by itself and has attracted researchers and industry for decades, OTC is by no means the only relevant approach. The following section gives an overview of related work and compares it with the OTC solution.

6.2 Related Work

The usage of computers to improve traffic control for urban road networks has been investigated since the 1960ies. Motorised traffic has large impact on the environment and economics which motivated both – scientific and commercial research – to develop novel solutions. The range of applications for computer-aided or automated systems includes off-line planning, reactive processing of control logic, and automated traffic-adaptive systems. Considering real-world installations in cities worldwide, centralised commercial systems are wide-spread. Centralised systems or static optimised solutions (which are quasi-standard) are characterised by inherent limitations: small deviations from the normal traffic situation can cause major delays due to static configurations from a global perspective or slow reaction times of central elements. Thus, self-organising approaches have been developed.

To introduce the state of the art in traffic control systems, the following part of this section gives an overview of different research directions. Initially, *centralised systems* are described accounting for the major part of the installations worldwide. Afterwards, a wrap-up of more recent approaches with a strong focus on *self-organising* aspects is given, although most of these examples have a more scientific and theoretical background.

6.2.1 Centralised Systems for Traffic Control

One of the first successful approaches towards automated traffic control is the **Split, Cycle and Offset Optimisation Technique** (Scoot), which has been developed in the 1980ies [200]. The system is installed at over 200 cities worldwide. Based on the computation of a single cycle time for all contained intersections, *Scoot* determines the best configuration of phase lengths for each intersection controller and then adjusts the offset times between neighbouring intersections in order to minimise waiting times. To be able to perform this adjustment, the network is split into sub-networks and a dynamic traffic model is used.

Besides Scoot, several other commercial traffic control systems have been developed. The **Sydney Coordinated Adaptive Traffic System** (Scats) [201] emphasises especially the regional character of the system. Instead of a fully centralised solution, regional entities are responsible for the strategic control of the intersections' traffic controllers. The local controllers at the intersections are used to collect data and perform the tactical control (e.g. extend phases based on arrivals). *Scats* itself relies on a rule base of different controls, which are selected according to traffic conditions. The system-wide optimisation of the

control strategy for the urban traffic network takes the current traffic state into account. During the night, the system might try to minimise the number of stops, while it tries to maximise the throughput at day time.

More recently, a third popular system has been developed: the **Traffic-responsive Urban Control** (TUC) system [212]. TUC focuses on the traffic-responsive control and coordination of traffic signals within large-scale urban road networks. Based on the initial work by *Diakaki* [211], the system has been extended and applied to testgrounds in Greece [213]. The main focus of the system is to determine appropriate cycle times and to coordinate the intersection control sequences in order to establish PSSs using centralised knowledge. Additionally, it can consider public transport priorities. The approach distinguishes between two main parts: 1) a centralised cycle control mechanism is responsible for adjusting the cycle time to the maximum saturation level and 2) a decentralised offset control algorithm coordinates the main phases of consecutive nodes. Finally, the resulting signal plans for each intersection are adapted by a public transport priority module in order to favour public transport vehicles.

Another popular development is **Balance** [214, 215], which aims at finding the best coordination of intersections by simultaneously considering a prioritisation of public transport. The system is based on a hierarchical architecture with a central component. This central component is responsible for determining the cycle time, the maximum and minimum green times of the traffic-responsive controllers, and the offsets between intersections by using a heuristic approach. Furthermore, the particular green times and the phase sequences are determined at each intersection controller using a simple microscopic model. The process to determine the best settings for the network is based on traditional traffic objectives like queue lengths, number of stops, and waiting times [216, 217].

Furthermore, subsequent systems have been developed which make no longer use of variables like *cycle*, *split*, and *phase sequence*; these systems are therefore called *acyclic*. Here, the green times of individual phases are modified directly. Several examples for such acyclic systems can be found in literature, the most prominent are the American system **Opac** [218, 219], **Cronos** [220, 221] from France, and the Italian **Utopia/Spot** system [222]. Since all these systems are centralised solutions, they have to use simplifications to keep the task of finding the best solution for a network-wide control strategy and coordination tractable. Among these, the *Utopia/Spot* system is the only one allowing for a network-wide optimisation by using a macroscopic model of the network.

Centralised traffic control systems are characterised by the following inherent problems. Although they are powerful, they are also complex and hard to configure. Besides these configuration aspects, they have been judged to have a limited traffic-responsive behaviour during rapidly changing conditions (see *Dinopoulou et al.* [213]). Especially disturbances in normal traffic patterns like unforeseen peaks or incidents affect the impacts of the control strategy. Furthermore, typical limitations for systems with central elements can be observed with the *single point of failure* as the most prominent one. Determining the traffic situation

has to be done locally at the particular intersections using physical detectors (e.g. *induction loops*) – this local knowledge has to be accessible from server side. With increasing network size, the amount of collected data to be transferred to and processed by the server is getting tremendously large. This requires high capacity in terms of computation power and communication bandwidth.

Even more promising is a (at least partly) decentralised operation. Every intersection controller can perform traffic-adaptive controls based on locally measured data only. Communication is taken into account if an additional coordination with local neighbours is necessary or if a centralised control element adjusts the control policy. Typically, an intersection on the one side of a network does not need information about what is happening on the other side. Thus, systems with locally organised subsystems have become more prominent.

6.2.2 Self-organising Approaches in Traffic Control

Since centralised solutions to traffic control have drawbacks, research shifted the focus towards the investigation of locally organised solutions. Already in 1986, *Barriere et al.* [223] presented a decentralised variant of the **Prodyn** traffic control system as introduced in [224, 225, 226]. *Prodyn* is inspired by studying the on-line computation of a short sample time optimal control of an intersection. Although the locally organised variant does not require any central components, the resulting performance describes a behaviour similar to the hierarchical variant. The authors state that the main advantages of their system are good scalability and enhanced reliability, but further documentation and progress of the system or dependable results of the concept have not been published or are not well-documented.

A more recent approach to locally organised traffic control has been proposed by *Helbing et al.* in [227]. They presented a fluid-dynamic model for the simulation of traffic networks [228], which is used to develop a self-organising control principle for traffic lights. Their switching strategy for traffic lights at intersections is based on the "pressure", which is generated by cars waiting to be served. The more cars and the longer they are waiting, the higher is the pressure to give the corresponding turning movement the right of way. Simultaneously, cars blocking subsequent road sections create a "counterpressure" when green times cannot be used effectively in the current situation. Based on the current pressures and counterpressures, those traffic lights to be active within the next time period are selected. Furthermore, this results in a dynamic composition of turning movements for each run of the process. The intersections are loosely coupled by monitoring their connecting road segments, but do not perform an explicit coordination. According to *Helbing et al.*, this loose coupling is sufficient to dynamically establish PSSs as some kind of emergent effect. *Lämmer* compared the performance of this basic model to coordinated FTC systems [229] and states that a positive effect in terms of decreased delay times can be observed.

Although the approach has some desired characteristics like being highly decentralised

and adaptive, it has also drawbacks. *Helbing* assumes that the existing infrastructure at intersections will be significantly extended. In order to estimate queue lengths and delay times per car as well as the counterpressure for outgoing sections, a large number of additional detectors is needed in combination with further computational effort to analyse and aggregate the measured data. Furthermore, the dynamic composition of turning movements for each time period is problematic. Users might have difficulties to accept the system's behaviour (they cannot understand dynamic reassignment from their point of view) or legal restrictions have to be considered (like preventing conflicts during phase changes). Additionally, the current system does not consider so-called *qualified conflicting traffic streams* (e.g. traffic streams going straight ahead cannot be combined with left-turning traffic from the opposite direction simultaneously).

Similar to *Helbing*, *Gershenson* developed an approach for **Self-organising Traffic Lights** (SOTL) [230, 231, 232]. Both concepts have in common that their traffic-responsive controllers are locally organised and take only figures like the number of waiting cars or the gaps between arriving vehicles into account when determining their control decisions. In contrast to *Helbing*, SOTL is more focused on traditional phase-based traffic control systems leading to strong similarities to e.g. uncoordinated NEMA controllers [132]. *Gershenson's* control decision relies on a counter κ for the number of waiting cars in front of each traffic light with each car weighted by its waiting time. This κ value is used to decide whether the corresponding traffic light demands to switch to green or not (by using a predefined threshold). Since a simple threshold-based switching decision can lead to fast switchings and an oscillating behaviour, several restrictions are implemented in the SOTL control method, e.g. to avoid the interruption of moving platoons or deadlocks caused by long platoons. In addition to the purely locally organised control logic, *Gershenson* describes the observation of coordination effects similar to those achieved by PSSs, although the system has no explicit coordination mechanisms. But these effects can also be caused by the simple network structure (Manhattan-type) and the highly artificial traffic generation pattern (cars leaving the network are immediately relaunched at the opposite side of the network). A comparative implementation of *Gerhsenson*'s approach and the OTC system within a realistic network simulation tool has been developed in cooperation with *Zechner* [233]. In this study, SOTL showed nearly the same performance as OTC-controlled FTCs, although SOTL uses more knowledge and performs significantly more adaptations. Future work will compare SOTL to OTC-controlled NEMA controllers, where SOTL's advantage of faster adaptation cycles and a more traffic-responsive behaviour should be minimised. Hence, both approaches are expected to result in a similar performance.

An agent-oriented approach to decentralised traffic control has been presented by *Bazzan* in 2005 [234], where intersections are modelled as *individually-motivated agents*. These agents define a distributed traffic signal system that has to be coordinated in a decentralised manner. Thus, the approach is mainly related to the decentralised coordination mechanism situated at Layer 3 (see Section 6.4.1). The intersection-control agents select their behaviour

from a predefined set of control strategies with each selection based on observed local events as well as on the results of so-called *coordination games* that are played among neighbouring agents. *Bazzan* demonstrated the approach by applying it to simplified scenarios, in particular an arterial road containing 10 intersections. Within the scenario, the intersection-agents have to choose between just two strategies during the simulation – they have the choice to take part in a coordination for one of the two arterial directions. This agent-based coordination is compared to a centralised solution that establishes the coordination based on detector data. From the results of the comparative simulations, *Bazzan* concludes that her agent-based approach has advantages in situations where the flow of traffic in the different directions is nearly equal, while both approaches show the same behaviour in strongly unequal conditions. Compared to the OTC approach, *Bazzan's* system is based on the existence of predefined strategies to choose from. Consequently, it is hardly flexible and has to be configured for all possible situations at design time.

Similar to *Bazzan's* work, further more artificial examples are known in literature focusing on virtual traffic lights (intersection control without physical traffic lights): *Cakar et al.* presented their simulation of an "Indian junction" [30], *Chaaban et al.* [235] use a trajectory-based approach and investigate dynamic replanning in the presence of disturbances, and *Vasirani and Ossowski* [236] developed a reservation-based intersection control mechanism. Further systems are referenced e.g. in [237].

In contrast to the OTC approach and the context of this thesis, all of these approaches are artificial and currently not applicable to realistic environments. Furthermore, most of them require the existence of new technical solutions (both at cars and intersection controllers) and additional infrastructure like detectors. Besides legal aspects, especially safety reasons and problems when running in parallel with existing solutions will make a technical realisation within a manageable time period unlikely.

Recently, a third way besides already applicable systems and more artificial scenarios emerged. The **Next Generation Urban Traffic Control** (UTC-NG) project has its focus on future solutions for urban traffic control systems and integrated traffic management systems. The basic assumption is that upcoming systems will rely on a broader range of information. Fixed (e.g. cameras) or mobile sensors (e.g. vehicle-mounted) as well as communicated data from neighbouring intersection controllers are assumed to be available at each intersection. Current installations such as SCATS [201] break down the controlled network into subsets of intersections for which optimal strategies can be calculated. As a result, traffic flowing into and out of areas formed by sets of intersections requires complex interactions and is mostly neglected. Besides locally organised intersection control based on policies, UTC-NG proposes a stigmergy model to address the complex interaction problem between these areas. All kind of information is provided to a middleware that serves as stigmergy communication basis. Details on how this stigmergy is realised are not given yet. Currently, two main aspects have been focused and demonstrated using simulations of Dublin's inner-city road network: a *collaborative learning approach* [238] and a *middleware*

solution [239]. In contrast to UTC-NG, the OTC approach does not focus on sensor-fusion approaches and works with already existing infrastructure.

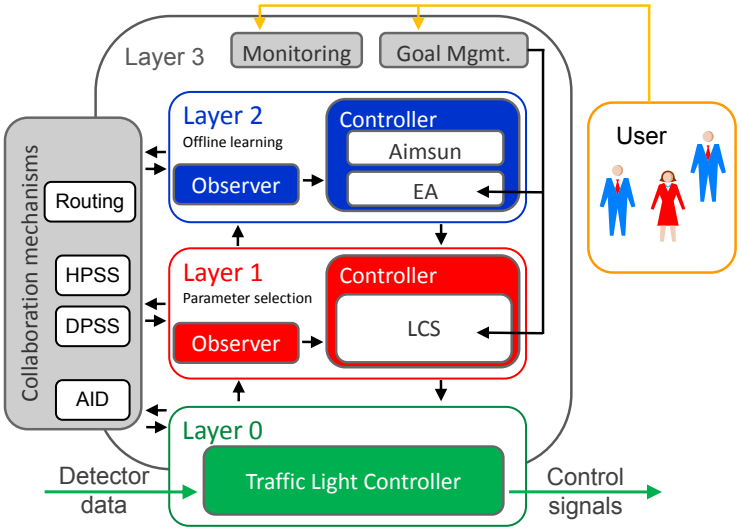


Figure 6.2: Design of the OTC system

6.3 Application of the Generic Architecture

The Organic Traffic Control (OTC) system is based on the general framework as presented in Chapter 3. In order to "wrap" traffic light controllers by the framework, a customisation of the general approach is needed – this customisation requires knowledge about aspects of the control process. Therefore, five basic tasks have been formulated to cope with the customisation:

- the definition of the environmental conditions and the node's status (the situation),
- a similarity metric for situation descriptions,
- a learning feedback to distinguish between good and bad behaviour,
- the definition of controllable variable parameters, and
- a simulation model (in combination with a simulation tool) for the "sandbox" component.

In the OTC architecture, an industry-standard *traffic light controller* (TLC) with fixed-time logic serves as SuOC. Alternatively, standard traffic-responsive controllers (e.g. following the NEMA standard [132]) can be used. This TLC is responsible for physically setting all of the intersection's traffic lights using a set of parameters that define its behaviour. The task for the OTC system is to choose these TLC parameters according to observed

changes in the traffic conditions at the particular intersection. The control loop consisting of Layer 1 and Layer 2 of the architecture is defined as follows. The information gained by the Layer 1 observer is passed to the Layer 1 controller, which is realised as an adapted Learning Classifier System (LCS) (see Chapter 4.1.4). In case where no appropriate rule is found, the Layer 2 component evolves a new parameter set for the specific situation. Therefore, an Evolutionary Algorithm (EA) is combined with the standard traffic simulation tool "Aimsun" [240]. Figure 6.2 depicts the adapted architecture for the OTC system. Details of the customisation are given in the remainder of this chapter.

(1) Situation Description

The observer of Layer 1 is responsible for monitoring the SuOC and its environmental conditions. Based on the description of the current situation at the intersection, the controller of Layer 1 has to decide whether an adaptation of the parameter configuration is needed or not. Thus, the *situation description* has to cover all information relevant to analyse the current status. For traffic control at urban intersections, the goal is typically to decrease delay times caused by red traffic lights and to increase the network's throughput. Detectors (like *induction loops* in the street surface or *video cameras*) serve as basis for determining the current situation – typically, they are used for calculating the number of waiting cars per traffic light, the flow (in $\frac{cars}{hour}$) for the corresponding turning movements, or the aggregated delay times at each turning movement.

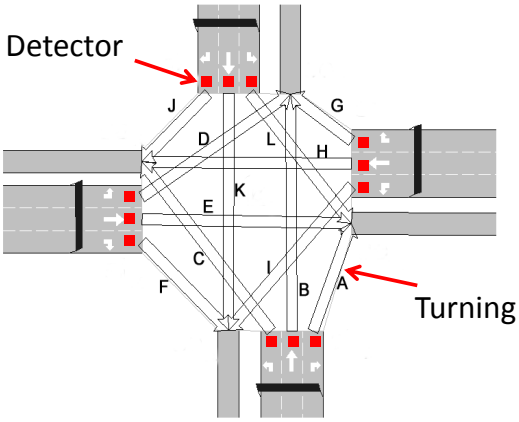


Figure 6.3: An examplary intersection with turnings and detectors

Figure 6.3 depicts an exemplary four-armed intersection. The turning movements crossing the intersection are labelled from A to L. Based on detectors situated at different points of the incoming sections, statistical data as outlined before can be determined. For OTC, the situation at the intersection is defined in terms of the measured flow-values per turning movement. For an intersection with n turnings, the system input consists of an

n-dimensional real-valued vector containing the traffic flows measured in vehicles per hour ($\frac{veh}{h}$) for each of the intersection's turnings. The flow-based situation description has been chosen, since the goal of the system is to find a balanced throughput for all streams passing the intersection. For instance, a situation description for the intersection from Figure 6.3 might look as follows:

Turning	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
Flow ($\frac{veh}{h}$)	500.0	60.0	10.0	50.0	95.0	33.0	98.0	10.0	10.0	10.0	10.0	10.0

(2) Similarity Metric

Based on this situation description, a similarity metric is needed to allow for a comparison of situations. The condition part of a classifier consists of n interval predicates forming an n-dimensional hyper-rectangle and the input situation is an n-dimensional vector. Thus, the Euclidian distance can be used as metric. If the value of dimension i is contained in the corresponding interval, the distance is 0. Otherwise, it is calculated in relation to the centre of the interval to avoid classifier conditions becoming too large.

(3) Learning Feedback

Several different metrics to quantify the performance of traffic control systems are applicable. Typically, the goal of a traffic engineers is to minimise the travel time through a network (or over an intersection), to reduce the number of stops per vehicle, or to avoid longs queues of cars waiting at turnings. Nowadays, metrics considering the environmental impact become more prominent: the vehicles' *fuel consumption* and their *emission of pollutants*. OTC provides a solution that can be applied to existing systems without huge additional impact by re-using existing infrastructure – hence, emission models and similar metrics cannot be considered as they are not observable with current technical installations. In traffic engineering, the performance of a controlled intersection is often measured in terms of the *Level of Service* (LOS) [241], which is the average delay per vehicle passing the intersection. For illustration purposes, this value is mapped to a discrete scale of six levels labelled A (no delay) to F (heavy congestion). Other measurements like number of *stops per vehicle* or *queue length* are sometimes incorporated into a performance index to represent optimisation goals. Due to its wide acceptance in traffic engineering, this LOS-value is used for the OTC system to quantify the system's performance. The formula is defined as follows [241]:

$$LOS(x) = \frac{\sum_{t \in T} f_t \cdot d_t}{\sum_{t \in T} f_t}, \tag{6.1}$$

where x is the particular intersection and T is the set of the intersection's turnings. The variables f_t and d_t denote the flow and the averaged delay for a turning $t \in T$.

(4) Configuration Space

The target of OTC is to adapt the traffic light control strategy according to the currently observed traffic situation at the intersection. Signalling at intersections in real-world installations has some characteristics that have to be considered when defining the configuration space of the OTC system. Figure 6.4 depicts an exemplary four-armed intersection with the corresponding signalling. Each incoming section has turning movements leading to the left, to the right, and straight ahead to cross the intersection. Typically, non-conflicting turnings are aggregated into signal groups, which receive the right of way together. The example of Figure 6.4 demonstrates such a grouping of turnings.

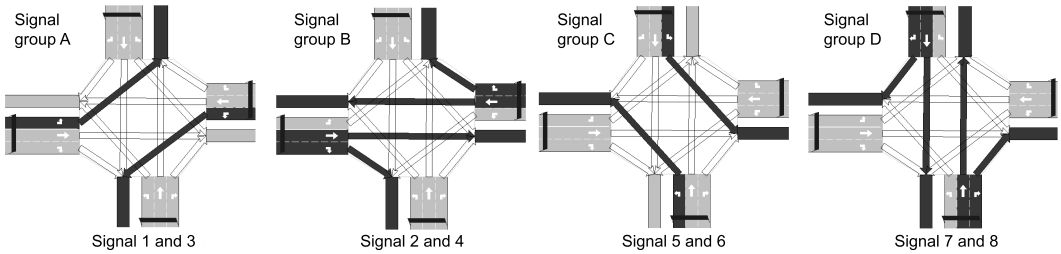


Figure 6.4: Exemplary intersection with signal groups

Figure 6.5 depicts an exemplary signal plan for the intersection from Figure 6.4, which corresponds to one possible action for the OTC system. The figure describes the green, yellow, and red times for each contained signal group (Signal 1 to 8) of the intersection. Within each cycle, each of these signal groups gets the right of way for a specific duration in a static and reoccurring order. The green boxes define the durations of having the right of way. Otherwise, the traffic light of this signal shows yellow or red (no box). The durations are defined using the scale on top of the figure. The ordering and the corresponding grouping of signals to signal groups is not altered by the OTC system – in contrast to the durations of the particular green times and the corresponding cycle time. Hence, one action describes the phase durations of each signal group (A to D).

(5) Simulation Model

Finally, the sandbox-learning principle of Layer 2 relies on using a simulation tool and a simulation model. In traffic engineering, several different simulation tools emerged starting in the mid 1980s with e.g. the *VISSIM* [242] predecessor *PELOPS/DYNAMO* [243]. Today, two of the most famous simulation tools are *VISSIM* [242] and *Aimsun* [129]. From

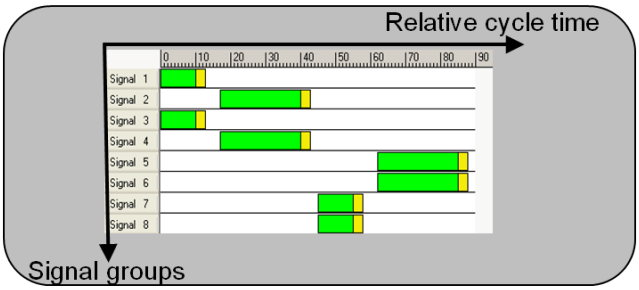


Figure 6.5: Corresponding signal schedule

these, the latter one is used within the OTC system, since it provides realistic simulations – a comparison of tools describing their advantages and disadvantages can be found in [244]. *Aimsun* is a software application for traffic engineers offering three types of transport models: *static traffic assignment tools*, a *mesoscopic simulator*, and a *microsimulation tool*. In the context of the OTC system, only the microsimulation for urban traffic networks is used. Alternatively to traffic simulators, approximation functions from the domain of traffic engineering like the one from *Webster* [245] can be used. Each intersection controller possesses an *Aimsun*-model of its topology taking only the local intersection into account. This model is configured with the observed traffic situation. The simulated traffic passing the intersection is equal to the observed one (in terms of the measured value in $\frac{cars}{hour}$ passing a specific turning movement). Based on this model and its configuration, an EA is applied to the optimisation problem. The simulator serves as a possibility to analyse the quality of the generated traffic signal plan for the given situation. Therefore, the *Level of Service* as introduced for the on-line learning component is used as fitness measurement.

6.4 Collaboration Mechanisms

The previous part of this chapter considered the OTC system as a single installation at one isolated intersection. Thus, only local measurements and locally observed data have been taken into account for deciding about the control strategy and the corresponding TLC's parameter adaptation. In contrast to such a strictly local focus, urban intersections in real-world installations are typically coordinated and integrated into a network-wide concept developed by traffic engineers. Consequently, initiatives in research and economy started to investigate integrated solutions typically referred to under the term "Intelligent Transportation System" (ITS), see e.g. [246]. Some of the responsibilities are already covered in older systems. For instance, establishing PSSs has already been covered by SCOOTS [200].

In order to provide the basis for such an ITS founding on the OTC system, Layer 3

of the architecture is introduced to handle collaboration among neighbouring intersections. In the context of the OTC project, several collaboration mechanisms have been developed which are introduced in the following part of this chapter:

- A fully decentralised mechanism to establish PSSs (the DPSS mechanism)
- A hierarchically extended version (the HPSS mechanism)
- A decentralised incident detection mechanism
- A traffic guidance system

The particular mechanisms are also depicted in Layer 3 of Figure 6.2.

6.4.1 Decentralised Progressive Signal Systems

Traffic networks in urban regions typically consist of a large set of neighbouring intersections. Considering such real-world traffic installations, the controllers between neighbouring intersections on main roads (like arterials) are often coordinated to form PSSs. Coordination can have a positive effect on the traffic flow, since the number of stops per car travelling through the network can be decreased. The goal of such a PSS is given as follows. Vehicles driving on a coordinated stream do not have to stop in front of traffic lights – they arrive at the intersection when the corresponding phase is switched to green.

Coordination depends on several factors. A first prerequisite for establishing a PSS is a *common cycle time* for all coordinated intersection controllers. Furthermore, each participating intersection controller needs to determine a *synchronised phase* starting always at a fixed point of the cycle. Since vehicles that are going to benefit from such a coordination have to pass a given distance between the neighbouring nodes, appropriate *offsets* have to be defined taking this travel time into account. This *offset* defines the relative start of the synchronised phase. Due to the static character of FTCs, coordination can be established by determining the previously mentioned variables. Although this task is more complex for traffic-responsive controller variants, it is generally possible [247].

The following part of this chapter presents a decentralised mechanism to establish PSSs. The mechanism has been initially presented in [247] and further discussed in [248, 208]. The approach distinguishes between three consecutive steps.

- Initially, all nodes in the network determine partnerships for collaborating in a PSS (Step D.1).
- Afterwards, nodes agree on a common cycle time which is a prerequisite for synchronisation (Step D.2).
- Once the cycle time has been chosen, every node selects its preferred TLC parameters with respect to the common cycle time and calculates the offset to its predecessor.

When the last participating node activated its new TLC, the PSS has been established (Step D.3).

Step D.1: Determine Collaborating Nodes

The first part of the DPSS algorithm is used to determine the participating intersection controllers. The goal is to find a sequence of intersections where coordination improves the network's traffic flow. A suitable heuristic approach when identifying appropriate streams without network-wide knowledge is based on determining the strongest streams at the particular intersections locally. Therefore, each intersection controller chooses its turning movement with the strongest vehicle flow. It is assumed that all intersection controllers of the network have *synchronised common clocks*, such that a periodic check is feasible where all intersection controllers perform the selection process at the same time.

In the following, the term *node* is considered to represent both, the *intersection controller* and the *corresponding intersection*. Imagine node j's turning movement from upstream node i to downstream node k is its strongest stream in terms of vehicles per hour. A synchronisation of the signal phase serving the corresponding turning should have the highest benefit for node j. If more than one signal phase matches this condition (serving the particular turning), the longest one is chosen since the phase duration corresponds to the relative flow values. After the selection and the corresponding creation of the *synchronised phase*, node j informs its desired predecessor (node i) about its intention to be its successor in a PSS. The process as described for node j is performed by all nodes of the network simultaneously – when all nodes informed their desired predecessor, each node performs a local matching. Within this matching process, each node compares the received information to its own preference. Matching predecessors are immediately acknowledged and partnerships are established. Other nodes that registered with node j are rejected.

Nodes receiving such a reject message have a second chance to find partners for a PSS. Therefore, these nodes select a new desired predecessor by choosing the turning with the second highest stream. The previous process is repeated with this second-best predecessor. Nodes with successfully established partnerships reject incoming messages, while nodes with open successor positions consider the request analogously to the first iteration. Since typical intersections in urban road networks are three- or four-armed, a second chance is enough for the process to find appropriate partnerships. All nodes becoming part of a PSS received acknowledgements, which allows for a self-organised determination whether the node is part of a PSS and which of its neighbours are predecessor or successor in the system. Nodes at special positions of the PSS (the *begin-* and the *end-node*) are aware about their position, since they have no predecessor (no successor) but a successor (a predecessor). Furthermore, isolated nodes being not part of a PSS did not receive or send acknowledgements and hence know about their isolation. Thus, partnerships are consistently established and nodes within a PSS can start to negotiate a common cycle time.

Step D.2: Determine a Common Cycle Time

In order to allow for establishing a PSS, the participating nodes have to agree on a *common cycle time* in the second step of the process. This common cycle time has to be chosen carefully, since it has direct influence on the node's capacity. On the one hand, a longer cycle time increases the fraction of green times as part of the cycle time, since the duration of interphases used to clear the intersection is always constant. If a smaller percentage of the cycle time is used for interphases, the capacity of the intersection is higher. On the other hand, longer cycles increase vehicle delays in undersaturated conditions due to the increased waiting times resulting from longer red periods. Thus, the cycle time determined by the intersection's OTC system provides a trade-off between these two aspects – it provides sufficient capacity while keeping delays short.

During step 2 of the process, a common cycle time is determined that fits these requirements. In order to allow for a fully decentralised determination, each node i has to keep track of two values: its own *desired cycle time* (DCT_i) and an *agreed cycle time* (ACT) for the PSS. The desired cycle time DCT_i is the currently preferred cycle time of node i if it is not part of a PSS. It can be received using the same process as when selecting actions to be applied to the SuOC. Furthermore, the agreed cycle time ACT is that cycle time the nodes taking part in the PSS agreed on. Since DCT_i of each node is as short as possible, ACT can be selected as the maximum of all DCT_i in the PSS. A shorter ACT could affect the most heavily loaded node of the PSS by decreasing its capacity and consequently inducing rising vehicle queues.

$$ACT := \max\{DCT_i\} \tag{6.2}$$

Determining the agreed cycle time ACT for the PSS is based on the following *echo algorithm* (cf. [249]). The first node in the PSS chooses its desired cycle time DCT_1, sets $ACT_1 := DCT_1$, and sends ACT_1 to its successor in the PSS. The succeeding nodes i, $i = 2, \ldots, n$ (with n the last node in the PSS), successively update their DCT_i by activating their LCS, setting

$$\begin{aligned} ACT_i &:= \max\{DCT_i, ACT_{i-1}\} && (6.3)\\ &= \max\{DCT_i, \max_{j\in\{1,\ldots,i-1\}}\{DCT_j\}\} && (6.4)\\ &= \max_{j\in\{1,\ldots,i\}}\{DCT_j\} && (6.5) \end{aligned}$$

and sending ACT_i to the next node in the PSS. The process terminates when reaching the last node n of the PSS. At this point, node n's ACT_n is already the sought-after one. Thus, it has to be propagated back to the predecessors, such that each node i in the PSS can update its ACT_i:

$$ACT_i := ACT_n \ ; \ for\ i = 1, \ldots, n-1 \tag{6.6}$$

The second part of this echo-algorithm terminates when node 1 is reached – all nodes taking part in the PSS have agreed on a common cycle time.

Step D.3: Determine Offsets and Establish Synchronisation

Finally, the third step of the algorithm establishes the PSS. Therefore, each node beginning with the first one has to select an appropriate TLC (with respect to the ACT) and to determine the *offset* to its predecessor. Since the first node does not have a predecessor, no offset restriction exists. Thus, it can directly choose and apply a TLC according to the ACT. To cover the constraint of choosing TLCs according to a given cycle time, OTC's learning component situated at Layer 1 of the architecture has been equipped with an additional procedure – details are given in Section 6.4.1. As a result, the OTC system selects a TLC that suits both – the current traffic conditions and the ACT. For each succeeding node i, $i = 2, \ldots, n$, the TLC is chosen accordingly – in addition, the particular offset o_i depends

- on the predecessor's offset o_{i-1},
- on the start p_{i-1} of the synchronised phase within the predecessor's TLC,
- on the time $d_{i-1,i}$ vehicles need to arrive from the predecessor,
- on the start p_i of the synchronised phase within the node's own TLC, and finally
- on the time q_i needed to serve queued vehicles for the synchronised phase.

In addition, node 1 has to communicate the absolute time s when activating its selected TLC, since all successors have to know the start time of the PSS to determine their relative start. Afterwards, a second *echo algorithm* is used to successively establish the PSS. The first node communicates the start time s, its offset (which is zero for the first node), and the start p_1 of the synchronised phase in its TLC to its successor. Based on this information, nodes i (with $i = 2, \ldots, n$) successively select their own TLCs (according to the ACT), determine their p_i, and calculate their own offset relative to the first node in the PSS using the formula:

$$o_i = (o_{i-1} + p_{i-1} + d_{i-1,i} - p_i - q_i) \mod ACT. \tag{6.7}$$

The underlying assumption of time $d_{i-1,i}$ being stored locally at each node for all its neighbours j is reasonable, since $d_{i-1,i}$ depends only on the fixed distance and speed limits between neighbouring nodes. Furthermore, q_i is currently chosen as a small constant, which will reflect characteristics of the particular connection and the local traffic in the future. After finishing the offset calculation, the values for s, o_i, and p_i are forwarded to the successor until the process terminates with reaching the last node of the PSS.

Temporary TLCs for Establishing the PSS

A further aspect has to be considered when establishing the PSS by finally changing TLCs. Due to clearing times and safety restrictions, an abrupt change of controllers is infeasible. Such a change might lead to undesired behaviour in terms of unbalanced green times. To counter these effects, a *temporary TLC* is activated for exactly one cycle after the currently active TLC's cycle has ended. This temporary TLC is used to close the gap between the currently active TLC and the desired begin of the new one. Therefore, all non-interphase durations of the currently active TLC are proportionally adjusted (typically decreased). Its cycle time t is given by the equation

$$t = s + o_i - r - c \tag{6.8}$$

where r denotes the remaining duration of the active TLC's cycle and c is the current time at the node. In those cases where time t is not applicable because it is too short (t is smaller than the required minimum duration for each phase plus all interphase durations), t is adjusted by ACT:

$$t := t + ACT \tag{6.9}$$

After finishing one run of the temporary TLC, the PSS is finally established by activating the selected TLC. The temporary TLC process explains a lower boundary for the DPSS-update interval. If it is performed too often, the learning process of OTC's Layer 1 component is disturbed, since the temporary TLC affects the performance of the OTC system and is not considered during the evaluation phase.

Time Requirements for the Communication

The previously introduced distributed algorithm requires synchronised clocks [250] available at all participating nodes. This assumption can be realised by using e.g. a GPS receiver or time synchronisation protocols such as the *Network Time Protocol* (NTP) [251]. Furthermore, Step D.3 of the algorithm assumes bounded processing times. A known upper bound π is required for the processing at each node. Furthermore, the communication latency between two neighbouring nodes of the PSS is restricted by a known upper bound λ. The necessity is caused by the need of switching new TLC configurations reasonably accurate at a given time in the future. In the context of the investigated scenarios, the start time s for switching to the new configuration needs to be at least

$$s > t_0 + (n-1) * (\pi + \lambda) \tag{6.10}$$

where t_0 is defined as the time at which the first node starts establishing the PSS and n is the number of intersections in the PSS. Typically, the time period between establishing two consecutive PSSs is significantly larger than the latency for communication and processing. Furthermore, the number of intersections n is assumed to be low. Consider as example $\pi = 0.1\,s$, $\lambda = 0.2\,s$ and $n = 10$; the new configuration can be established in $9*(0.1\,s+0.2\,s) = 2.7\,s$ or later after starting the third step of the algorithm. Since TLC parameters are changed every 15 minutes on average, a time period of $2.7\,s$ can be neglected. Hence, the communication requirements are met even in networks with comparably high latencies.

Updating a PSS

Besides initially establishing a new PSS, a *dynamic update* is needed due to the time-dynamic nature of traffic. Instead of a complete recalculation in a reoccurring sequence and independently of the changes in the traffic conditions, the system reacts on the following events representing relevant traffic changes:

- A node participating in a PSS changes its TLC due to a (significant) improvement of local delays predicted by its LCS. The new TLC respects the agreed cycle time ACT of the PSS, but the start time of its synchronised phase differs from the original TLC.
- A node i participating in a PSS (significantly) increases its desired cycle time due to changes in traffic such that $DCT_i > ACT$.
- A node i with $DCT_i = ACT$ (significantly) reduces its desired cycle time due to changes in traffic.
- For a node participating in a PSS, the turning exhibiting the strongest vehicle flow has changed.

In the first case, only the changing node i is concerned: it has to adapt its offset such that the new offset o_i' satisfies the equation

$$o_i' = (o_i + p_i - p_i') \mod ACT \tag{6.11}$$

where p_i' denotes the start of the synchronised phase in the new TLC, while o_i and p_i denote the respective values for the previous TLC. The established partnerships, the agreed cycle time ACT, and the offsets of all other nodes participating in the PSS may remain unchanged.

In the second and third case, the agreed cycle time ACT needs to be adapted, which is achieved as follows. The node responsible for the change announces the necessity of an ACT update to the first node in the PSS, which as a result starts the *echo algorithms* for determining a common cycle time and offset recalculation resulting in an updated PSS with

the same collaborating nodes. Since the update temporarily reduces the performance of the collaborating nodes again, the process should only be activated if the change in the agreed cycle time ACT exceeds a given threshold.

Finally in the fourth case, the importance of traffic movements has changed, while in the other cases only traffic volumes were responsible for changes. Here, the partnerships in the PSS should be reassessed by completely recalculating new PSSs.

Integration of DPSS into the OTC Architecture

The DPSS algorithm is embedded into the OTC system at Layer 3 (see Figure 6.2) of the architecture. In addition, the learning component of Layer 1 has to be able to handle the additional cycle time restriction that is necessary when selecting TLCs under constraints given by a PSS. In particular, the selection procedure of the learning component realised as a modified LCS has to cover these cycle time constraints by restricting the set of possible solutions to those proposing appropriate cycle times. Therefore, the initial selection process is adopted by considering a further condition. If none of the rules matching the traffic conditions proposes an appropriate cycle time, rules (closely) matching the traffic situation are modified to fit the cycle time restriction by proportionally decreasing or increasing the non-interphase durations of their actions. This modification of rules is based on copying existing ones and including them in the rule set – similar to the presented *covering mechanism*. This extension leads to a full integration of the DPSS algorithm into the existing OTC approach.

6.4.2 Hierarchical Progressive Signal Systems

Figure 6.6: Traffic flows in a Manhattan-type network

The decentralised process as presented before selects the partners for the PSS based on the locally observed strongest turning movements. As a consequence, the strongest streams within the network are preferably treated – which is a good heuristic, but it does not always establish the best possible PSS configuration for the network. Figure 6.6 illustrates such a non-optimal example.

The Manhattan-type network of the figure contains six intersections and five prominent traffic streams. Two streams are running from west to east, while three streams are running from north to south. The width of the arrows is proportional to the corresponding traffic flows for the depicted streams, traffic for the other directions is neglected in this example. The strongest stream is the central one of the southbound streams, but in sum both east-

bound streams are larger than the three streams running from north to south. Considering the DPSS mechanism's algorithm as presented previously (in particular Step D.1), the approach would initially establish a PSS-coordination for the central southbound stream as the strongest stream. Further iterations of Step D.1 would also result in a coordination of the other southbound streams. A comparison of arrows' widths in Figure 6.6 demonstrates the non-optimal character of this solution. In total, the establishment of PSSs for the two eastbound streams would lead to a higher benefit.

In order to be able to decide automatically about favouring a set of PSSs to others, the benefit of the synchronisation has to be measurable. Less formally, the benefit of a PSS can be defined as number of cars having the advantage of not stopping anymore due to the coordination – a vehicle driving on a coordinated stream can pass the intersections without having to stop at a red light. The coordination affects all intersections taking part in a PSS except the first one, since vehicles arrive randomly. As a consequence, the sum of all vehicle flows for the coordinated turning movements at all intersections except for the first one represents the desired benefit. This issue is expressed by the following formula:

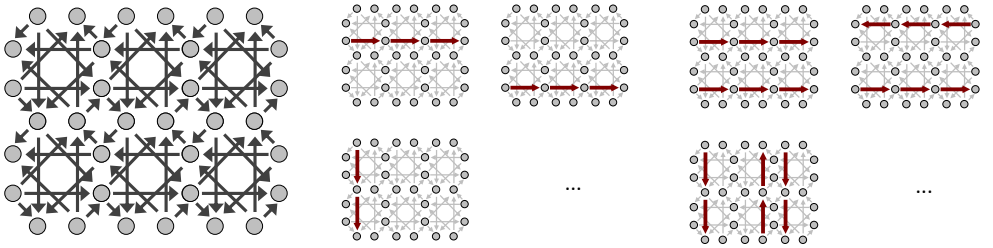
$$benefit(PSS) = \sum_{i=2}^{n} (synchStream_i) \tag{6.12}$$

with n being the number of all participating nodes in the PSS, i specifying the particular node, and $synchStream_i$ the traffic flow of the synchronised stream at node i.

The following part of this chapter extends the DPSS mechanism with an additional hierarchical component – the Regional Manager (RM) [252, 208]. The RM is capable of dealing with special cases as discussed in the motivating example. The approach is able to find better combinations of PSSs for the underlying network and its current traffic status than the DPSS mechanism – if available. In particular, it replaces Step D.1 of the DPSS mechanism, while keeping Steps D.2 and D.3 to finally establish the coordination. The process of finding *better* solutions emphasises the heuristic character of the RM – although it aims at determining the best possible combination of PSSs for the network, it does not compare all possible solutions. Thus, the optimality of the solution cannot be guaranteed. The RM works in three steps:

- **RM.1:** Collect local information about the traffic situation and build a weighted graph representation of the network.
- **RM.2:** Identify possible streams to be coordinated by PSSs.
- **RM.3:** Combine non-conflicting streams to stream systems and select the most promising stream system.

Figure 6.7 illustrates the concept. All three steps are discussed in more detail in the remainder of this section.



(a) Step RM.1: Build network graph

(b) Step RM.2: Determine traffic streams

(c) Step RM.3: Determine stream systems

Figure 6.7: Steps performed by the Regional Manager

In the following, a centralised element (the RM) is assumed to perform the computational tasks. Since no further system-wide information is needed besides communicated data of the participating nodes, the mechanism can also be realised in a distributed manner. Therefore, concepts from the domain of distributed systems like *Leader Election* [253] provide the possibility to select one node as responsible for the task. Such distribution variants do not have any influence on the mechanism itself and can therefore be neglected.

Step RM.1: Build Network Graph

Step RM.1 of the process derives a graph representation of the current traffic flows within the network as basis for the calculation of the succeeding steps. Therefore, each node creates a subgraph representing its topology and the local traffic flows. This subgraph contains one vertex for each outgoing section, one vertex for each incoming section, and one edge for each turning movement. Edges are directed and weighted, with the weight corresponding to the currently observed traffic flow of the turning. The RM receives the subgraphs from all nodes of the network, connects them, and consequently obtains a graph representation of the network augmented with the current traffic situation. Figure 6.7(a) depicts the result of Step RM.1 for an exemplary Manhattan-type network of six nodes (similar to the one from Figure 6.6).

Step RM.2: Determine Traffic Streams

Step RM.2 identifies promising streams where establishing a PSS can be beneficial for the network. Based on the graph $G = (V, E)$ created in Step RM.1, the RM iteratively builds streams by connecting edges $e \in E$ to candidate streams until all edges have been removed from E or the remaining edges' weights are below a predefined threshold. Therefore, it selects the edge e_i with the highest weight and iteratively determines the best predecessors and successors by selecting the edges with highest weights from the particular candidates. When continuing in both directions, the process of adding a new edge to the stream terminates

as soon as there are no further candidates or the next candidate has a conflicting higher stream within the current subgraph. The latter case avoids choosing streams according to the example in Figure 6.6: *lower streams will not cut larger ones.* Figure 6.7(b) depicts some of the resulting traffic streams obtained for the examplary network (Figure 6.6), each stream is visualised as a connected sequence of thick edges within the figure.

Step RM.3: Choose the Best Stream System

Finally, Step RM.3 is used to find the most-promising combination of streams. Thus, the selected streams need to be non-conflicting – in particular, they must not intersect each other or run in different directions on the same roads. As a consequence, each node can participate in one PSS only. The selection is based on the metric as initially presented, i.e. the combination of streams maximising the number of benefiting vehicles is searched. Therefore, the RM uses a greedy-approach to create promising stream systems without generating the power set of streams. Initially, potentially conflicting streams are identified and cached. Afterwards, the streams are ordered according to their potential benefit. Based on this ordering, stream systems are iteratively built by adding the next beneficial and non-conflicting stream. Furthermore, the process ensures that each stream is part of at least one stream system. Figure 6.7(c) depicts several systems of non-conflicting streams for the examplary network. Streams are again visualised as connected sequences of thick edges. The resulting set of different stream systems can be rated by summing up the benefit of the contained PSSs. Finally, the system with the highest benefit is applied to the network.

All steps performed by the RM can be efficiently implemented, details about the particular algorithms can be found in a previous publication [252]. With replacing Step D.1 of the DPSS mechanism by Steps RM.1 to RM.3, a hierarchical system architecture is obtained having similarities with the design of e.g. BALANCE [215]. The main difference to these approaches from literature is the locally organised selection and optimisation of signal plans.

6.4.3 Further Collaboration Mechanisms

Establishing coordination between neighbouring intersections using PSSs is a reaction on the observed status of the traffic network. In contrast, the objectives of upcoming integrated traffic management systems are broader. Traffic through the network has to be optimised by an active management of traffic flows. One aspect of such a broader view is an infrastructure which is actively guiding drivers to prominent directions. Such a concept relies on the availability of further knowledge about the network's status. Examples are an automated incident detection and the prediction of traffic flows' states for the upcoming time period. Thus, the following part of this chapter outlines two further collaborative mechanisms, which have been integrated into the OTC system: *incident detection* and *route guidance. Traffic*

prediction for OTC has been investigated in cooperation with *Volhard* and presented in [254].

Distributed Incident Detection

Automated Incident Detection (AID) is a prerequisite if the distributed routing protocol has to be capable of dealing with disturbances. In literature, incidents are defined as "events which cause a need for assistance of involved drivers and/or warning of oncoming traffic in order to maintain safe driving conditions" [255] meaning it is not only restricted to car accidents. In contrast, road blockades, unscheduled maintenance, and construction activities or breakdowns and spilled loads are also considered [256]. Typically, incidents are claimed to be a major reason for undesired effects like decrease of road capacity, increase of delays and pollution, congestion, and increasing cost [256, 257]. To counter these effects, a research area has been established investigating solutions to detect such events fully automated or at least machine aided. Existing approaches analyse data obtained from regularly distributed road sensors (like induction loops) and try to characterise the observed traffic conditions as incident-free or -bound. Hence, AID systems are not able to directly detect incidents (like it might be possible one day by computer vision), but indirectly through classifying traffic conditions according to known patterns. These patterns describe incidents as temporal and spatial road obstacles creating changes in the traffic flow. Besides the pure pattern recognition approach, several other techniques have been developed – a classification and survey of these approaches is given by *Martin et al.* in [258].

Currently, AID mechanisms are only applied to outer-city main routes like highways. A famous example in Germany is the "Autobahn NRW" approach [259] where the status of the motorway network in the Ruhr-area is observed and classified in terms of capacity utilisation and corresponding traffic jams. An application of AID algorithms to urban areas is significantly more complex, since incident-like traffic patterns can also be caused by normal activities. For instance, waiting queues in front of red traffic lights will cause the same pattern like stop-and-go traffic on highways. Furthermore, waiting vehicles dropping off a passenger or quickly delivering goods cannot be distinguished from real incidents when using existing techniques. The problem becomes even more complex as OTC builds upon existing infrastructure – consequently, an AID approach for Layer 3 has to work on induction loops only.

Thus, a modified variant of the *California algorithm* (CA) [260, 261] has been developed by *Klejnowski* [262], which is explicitly designed for urban areas and copes with the previously mentioned restrictions. The CA has been chosen as basis since it works on the currently available infrastructure conditions and is referred to as quasi-standard solution in literature. The algorithm detects incidents by *continuously comparing detector values* of two consecutive sensors. Thereby, one sensor represents two induction loops installed in the street surface together. The most significant difference to the initial application domain of

the CA is given by the signalised intersections. Comparing the measured flow values of the corresponding traffic flows passing these detectors would constantly result in false-alarms. In contrast to detectors on highways or non-interrupted roads, the problem might be handled by a cumulative strategy: an alarm is only given after a certain number of consecutive incident detections. But this strategy results in a significantly increased rate of non-detected incidents and still has a high false-alarm ratio. The modified CA addresses these problems by introducing a collaborative solution between neighbouring intersection controllers. In this context, *neighbouring* means that a direct connection between two intersections (both sharing the same road) is needed. In addition, the distance between these intersections is not allowed to exceed a certain length. The collaboration is based on a plausibility check for the detected alarms taking the incoming and outgoing traffic of the connecting road into account. Since an intersection receives the outgoing traffic from its predecessor, it can verify the predecessor's alarms. Such a verification is only applicable, if the traffic patterns are comparable – this leads to a maximum distance between nodes. In the conducted experiments, neighbouring intersections with a distance of about $250\,m$ had a good performance (about 100 % detection rate and below 5 % false alarm rate depending on the scenario), while distances larger than $1\,km$ showed substancially worse results. In contrast, the standard CA leads to significantly worse results – in some cases, more than 50 % false alarms by only 75 % detection rate. Details on the algorithm and further results can be found in [262].

Distributed Traffic Guidance

In contrast to investigating traffic signalling and coordination mechanisms, *traffic-responsive route guidance* is a relatively young research area. Most of the installations in real-world traffic systems rely on the usage of *Variable Message Signs* (VMS). These VMS are used to communicate route recommendations to drivers and are typically controlled by human-operated traffic control centres. Besides these often motorway-bound VMS, a more fine-grained routing recommendation system is known by navigation systems. Early navigation systems have performed the route guidance task based on static information about the road network, while nowadays the market share of more traffic-responsive systems is increasing. These systems take additional data into account received by the radio's *Traffic Message Channel* (TMC, available mostly for highways) or on-line travel time databases provided by system manufacturers. These databases contain information about travel times recently observed by other drivers using the particular navigation system. In addition, a novel approach emerged previously introducing *floating car data* (FCD). This FCD can be used to analyse the current traffic situation within the road network. One example is the honey-bee-inspired *BeeJamA* approach as introduced by *Wedde et al.* [263].

The OTC system has been extended with a VMS-based approach [208]. Signalised intersections are assumed to be equipped with VMS at their approaching sections. These

VMS provide the possibility to communicate information about the currently best route choice taking some prominent destinations into account. The recommendation is based on informing about which turning to take and how long it will approximately take to the destination. Therefore, each intersection controller contains a routing component, which is responsible for determining route recommendations to these prominent destinations. These recommendations are stored in *routing tables* for each incoming section. One aggregated routing table for the intersection controller is not feasible, since the recommendations of incoming sections differ. Drivers are not allowed to turn on the intersection and come back on the same road as they approached the intersection. Figure 6.2 illustrates the integration of the mechanism into the architecture of the OTC system. In principle, this routing component can incorporate varying routing protocols – the variant presented in the following is inspired by the *Distance-Vector Routing* (DVR) [264] as known from the Internet.

Routing of vehicles in road networks differs fundamentally from routing in data communication networks. Besides missing acceptance of high delays, mechanisms like buffering of elements, resorting, or discarding elements (in data communication an element is a *packet*, in traffic a *vehicle*) are not feasible. Hence, the original DVR protocol has to be adapted to match the requirements of traffic control. The modified approach works as follows. At the beginning, each intersection controller checks whether it has direct access to a prominent destination by a connecting section. In this case, the controller generates an entry for the routing table of each incoming section that has a turning movement to the corresponding outgoing section leading to this direction. These entries store the destination, the recommended turning, and the currently predicted time to reach the destination. The latter value can be determined by receiving the current delay at the corresponding turning movement and adding the (static) travel time for the outgoing link. The delay is calculated based on the current traffic flows at the intersection and the turning's green time fraction using a standard formula from the field of traffic engineering (the *Webster* formula [245]).

As soon as a routing table entry is created or updated, the contained information is submitted to the corresponding upstream neighbour. The receiving neighbour adds the travel time needed to reach the sender to the submitted value of the message. Afterwards, the routing tables of all incoming sections with a turning connection to the sending neighbour are checked. A new *routing entry* is generated from the received information for previously unknown destinations. Otherwise, the existing routing entry is checked. In case this routing entry recommends the same turning to the sender, the entry's travel time is updated with the new one. In case of the existing entry recommending a different turning movement, the travel times are compared and only the entry predicting a shorter travel time is kept.

The previously described steps are processed by all intersection controllers similarly. As a result, routes to the initially determined destinations are iteratively propagated through the network. By performing the DVR-inspired routing protocol, a decentralised mechanism to guide traffic through the underlying traffic network without the need of a centralised

element is performed. In combination with mechanisms to detect incidents in urban road networks, the approach allows for a traffic-responsive routing in the presence of disturbances and enables robustness in traffic control. The protocol can be augmented by supporting an intra- and inter-regional routing, which reduces the effort in terms of computation and communication and consequently increases the scalability (see [265]).

In comparison to the previously mentioned state of the art, the approach has significant advantages. In contrast to the FCD technique, it does not require equipped vehicles. Radio's TMC has the disadvantage of communicating all known traffic information (or disturbances), while the VMS approach can be organised using a hierarchical indication. Destinations far away are aggregated into regions ("to *district A* turn left, duration 15 to 20 minutes"), while nearby destinations are explicitly mentioned. Furthermore, the direct access to the observed traffic situation leads to more up-to-date information and thus a better description of the network's current status. This effect is even stronger supported by the reduction of communication to and aggregation by a third entity (e.g. the manufacturer of navigation systems). In the standard DVR protocol, some major problems are known – the *count-to-infinity* problem [266] is probably the most famous one. In the modified variant, this problem does not occur due to two reasons: a) typically, traffic situations do not change abruptly and therefore link cost are only slightly changing over time, and b) the approach relies on recalculations of the network's complete list of routing tables, which avoids using outdated information.

Similar to the DVR-inspired approach, *Lyda* developed three further routing mechanisms and integrated them into the OTC system [265]: a *greedy-based* approach, an approach founding on the *A-star* algorithm [267], and a mechanism inspired by the *Link-State Routing* (LSR) protocol [268] as known from the Internet. In addition, a region-based extension similar to the Internet's *BorderGateway* protocol [269] has been developed for the DVR and LSR algorithms resulting in a better scalability of the approaches.

6.5 Evaluation

The experimental evaluation of the OTC system distinguishes between three scenarios: an *artificial Manhattan-type* road network, a model of the *stadium environment* situated at Hannover, Germany, and a model of an *intersection of an inner-city suburb* situated at Hamburg, Germany. The first example is based on an artificial road network with manually optimised and configured reference values, while the latter two examples represent the current installations as developed by traffic engineers. In addition, both networks from Hannover and Hamburg are configured using realistic traffic data from a census and the actual signallings. All experiments have been performed on a Dual Core PC running at 1.66 GHz each and 4 GByte RAM. The Aimsun simulator has been used in version *AIMSUN NG Professional Edition 5.1.11* based on a *Windows 7 Professional 64-bit* operating

system.

The evaluation demonstrates the benefit of using OTC in disturbed and undisturbed situations. For comparison purposes based on taking the performance obtained by the current installations into account, representative metrics are needed. As discussed when defining the local evaluation function of the learning mechanism, OTC's goal is to reduce waiting times occurring locally at the intersections. These are measured as averages based on the queues of all incoming sections. The simulation-based evaluation allows for calculating a mapping between driving behaviour (driving, waiting, stopping, and accelerating) and the corresponding fuel consumption by taking realistic models into account [270, 271]. In addition, the network-wide view allows for evaluating the averaged number of stops per vehicle – which are going to be reduced by mechanisms like PSSs. Thus, the traffic-based analysis of OTC's performance is based on these basic metrics (delays, fuel consumption, number of stops) in the remainder of this chapter.

6.5.1 An Inner-city Area at Hamburg, Germany

OTC has been developed as a novel approach in traffic control. Thus, its inherent goal is to provide a substantially better solution than existing approaches in traffic engineering. Since a comparison to industry-standard solutions is not feasible at urban installations due to proprietary, monetary, legal, and safety-based reasons, a simulative evaluation has been performed. The system provides an applicable solution for real-world traffic control – hence, a comparison to existing control strategies according to the actual traffic conditions is needed. Therefore, the topology of inner-city regions has been modelled in the Aimsun traffic simulator taking realistic traffic data from a census and the actual control strategies of the particular intersection controllers into account. Figure 6.8(a) depicts the investigated road network situated at Hamburg, Germany.

(a) The road network at Hamburg (b) Model of the intersection

Figure 6.8: Investigated road network located at Hamburg, Germany

The traffic data and the signal schedules are provided by the *Schmeck Ingenieurge-*

sellschaft mbH, Hamburg, based on a manual optimisation performed by order of the *Freie und Hansestadt Hamburg*. The corresponding traffic data has been obtained by a census processed at Tuesday, May 4, 2004 and illustrates the typical course of a day without further irregularities. In this census, cars and trucks passing the intersections were counted and documented for each turning with a time resolution of 15 minutes. Fixed-time signal programs for all nodes of the network have been developed by traffic engineers. These programs are used in reality and serve as reference controllers in the evaluation.

Without activating collaboration mechanisms of the architecture's Layer 3, the OTC system is an isolated installation at each intersection. Thus, the first part of the evaluation focuses on an exemplary single intersection to demonstrate the potential benefit of the OTC system. Figure 6.8(b) depicts the topology of the intersection.[2] The corresponding Aimsun model has been configured with the traffic data obtained by the census and the fixed-time signal program used in reality. The OTC system is assumed to start without further knowledge. The initial population is empty, only the standard signal program is known in advance. Figure 6.9 depicts the traffic demand during the day. The y-axis depicts the number of vehicles (cars and trucks) passing the intersection. The following results are obtained as averages from five runs of the particular simulations with different random seeds.

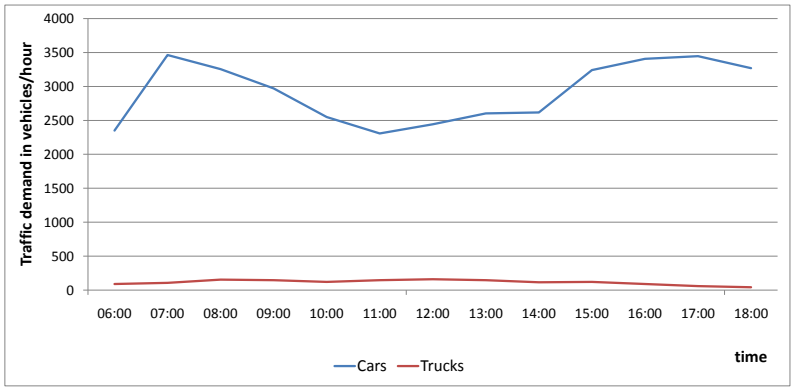


Figure 6.9: Traffic demand in number of vehicles passing the intersection

The main goal of the OTC system is to decrease the occurring delays at the intersection. Figure 6.10 illustrates the achieved results for three consecutive days in comparison to the reference solution. The x-axis depicts the simulated time of the day (6 am to 7 pm: 13 hours), while the y-axis shows the measured averaged delays for all turning movements. The lower the delays are, the better is the corresponding control strategy. Obviously, the reference solution has been outperformed at all three days. Compared to the reference solution, OTC reduced the occurring delays at the intersection by 14.1 % at day 1. Day 2

[2]The intersection as depicted in Figure 6.8(b) is located in *Hamburg-Eppendorf* and consists of the streets *Alsterkrugchaussee*, *Deelböge*, and *Borsteler Chaussee*; the figure shows the Aimsun model.

has been slightly better with a decrease of 15.6 % and day 3 stayed almost at the level of day 2 (15.9 % reduction).

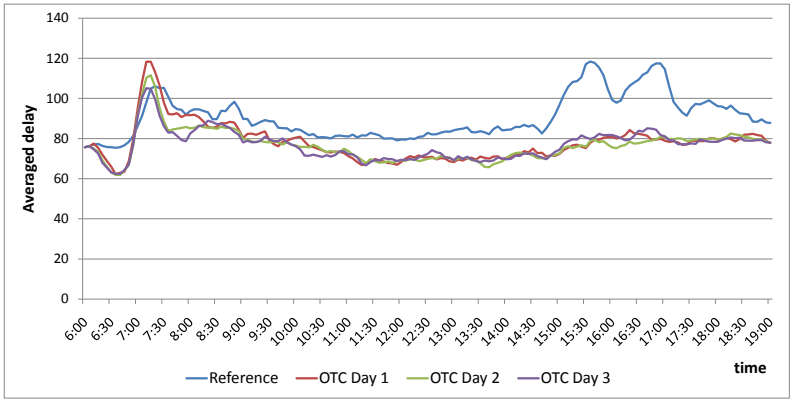


Figure 6.10: Averaged delay for three consecutive days (lower values are better)

Besides the pure delays occurring at an intersection, traffic engineers try to decrease the number of stops per vehicle. Typically, this metric is obtained at network level since it describes the coordination effect achieved by e.g. establishing PSSs. Besides the coordination effects, the metric is influenced by the control strategy since a balanced configuration of the phase durations will tend to increase the probability to cross the intersection without stopping. Figure 6.11 depicts the obtained results. The reference solution resulted in an averaged number of stops per vehicle of 2.41 *stops*, which is caused by the high traffic demands. This value has been reduced by the OTC system to 2.32 *stops* at day 1 (3.67 %). The following two days underline the results of day 1 (day 2: 5.09 %, day 3: 4.51 %). The increase at day 3 is within tolerated simulative deviations.

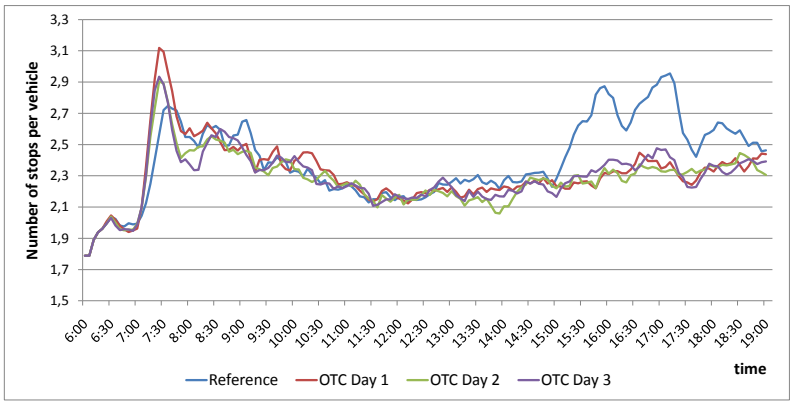


Figure 6.11: Number of stops per vehicle (lower values are better)

The reduction of delays and the number of stops are the most visible aspects for drivers passing the intersection regularly. A more indirect measurement is the *fuel consumption*,

which is significantly influenced by waiting and stopping at traffic lights. Especially in the context of current debates about the environmental impact of traffic, the reduction of fuel consumption is an important issue. Therefore, the environmental models integrated in Aimsun have been used to evaluate the vehicles' fuel consumption. The models consider idling, accelerating, and decelerating periods of the simulated vehicles as well as different cruising speeds. Their configuration is according to AIMSUN's manual using data from 1992 and 1994, respectively [270]. In general, more current figures might affect the overall consumption figures, but they would not change the proportional reductions. Figure 6.12 depicts the results obtained from the simulations. The reference solution results in an averaged fuel consumption of 15.23 $\frac{litre}{km}$. Due to the OTC control, this value can be decreased significantly. During day 1, the fuel consumption is reduced by 5.18 % to 14.44 $\frac{litre}{km}$ (day 2: reduced by 5.84 % to 14.34 $\frac{litre}{km}$ and day 3: reduced by 5.77 % to 14.35 $\frac{litre}{km}$). The positive effect is already visible at day 1, while day 2 and day 3 lead to nearly the same results. In general, an overall reduction of nearly 6 % just due to the organic control of one isolated intersection at normal traffic conditions is an auspicious result.

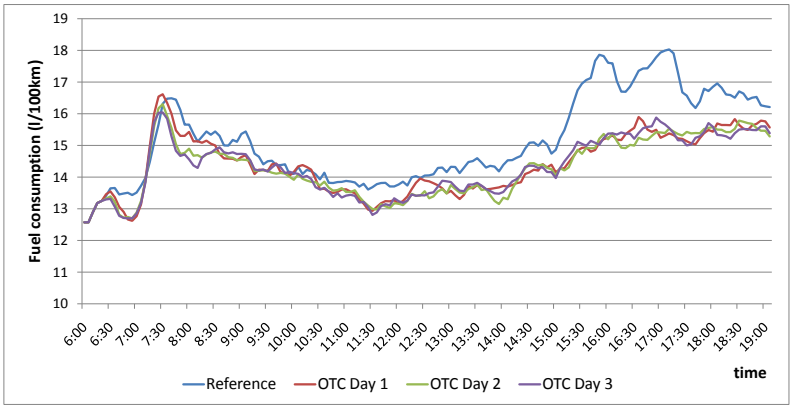


Figure 6.12: Fuel consumption (lower values are better)

Adaptation Effort of the Framework

The previous results demonstrated the benefit of using OTC in terms of traffic-based metrics. Another important aspect to be analysed is the performance of the framework itself – in particular, the adaptation and rule generation effort needed to achieve these results is a major criterion in the analysis. Therefore, Figure 6.13 depicts the progress of classifier generation using the Layer 2 component with ongoing simulation duration. Since the system starts with an empty population at day 1, no knowledge is available and the Layer 2 component is triggered constantly. This effect decreases significantly after the first day. During the first day, an averaged number of 90 new classifiers has been evolved. At day 2, Layer 2 evolved only 22 new classifiers on average and at day 3 only 15.3. This trend is

continued afterwards. Starting at day 4, maximally one new classifier is generated per hour. In addition, the probability of needing a new classifier decreases to nearly zero from day 10 on. Here, the rule base is almost static. The randomised traffic generation can lead to seldom deviations in demanding new classifiers. Thus, the rule generation converges after only a few days. In case of the occurrence of unknown situations, the population size could be further increased – but the *normal* classifier demand is covered with this population.

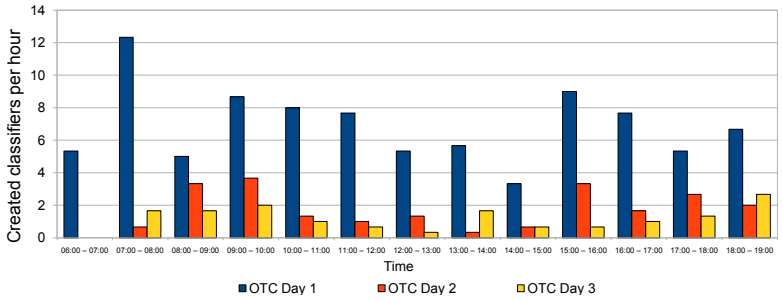


Figure 6.13: Development of the created classifiers in the Hamburg scenario

Besides the rule creation, the usage of classifiers is of interest, since it quantifies the adaptation demand identified by Layer 1. Figure 6.14 depicts the required number of classifiers at the first three simulated days. On average, 13.8 classifiers have been used per hour: maximally 18.7 and minimally 5.8. The profile during the day corresponds to the occurring traffic flows – a higher necessity of adaptation occurs in more dynamic situations. The number of possible adaptations is bounded by the sampling rate of Layer 1. Each parameter set has to be processed for at least two complete cycles to enable a qualified feedback. The cycle time of the created classifiers changes according to the particular traffic situation. In case of high demands (e.g. morning peaks), they tend to be longer while they are significantly shorter in case of low traffic demand (e.g. between 6 and 7 o'clock). For this intersection, the created classifiers recommend a cycle time of 91 s on average. Thus, the maximal number of adaptations per hour is given by: $3.600\,s/(91\,s * 2) = 19.8$ adaptations. As depicted in Figure 6.14, Layer 1 took advantage of most of the possibilities to adapt the SuOC's parameter set. This observation is caused by the highly dynamic traffic profile and will be different in cases of static or only slightly changing profiles. Thus, the effort spent on achieving OTC's adaptive behaviour is within an acceptable range – especially after the system's startup period, the rule generation effort decreases dramatically. In contrast, the adaptation process of Layer 1 is performed continuously – but performing the control loop every 3 min does not lead to unacceptably high computational effort.

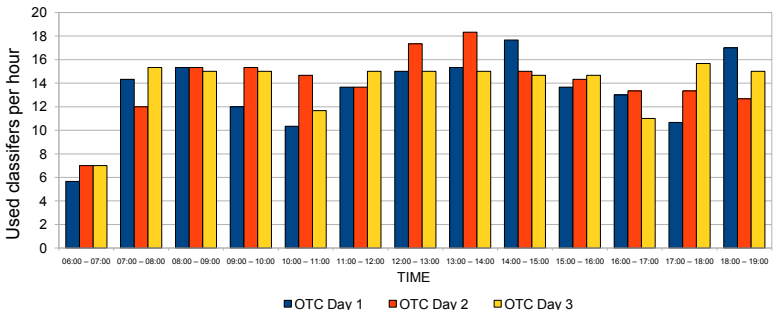


Figure 6.14: Adaptations of the SuOC performed by Layer 1 in the Hamburg scenario

6.5.2 The Stadium Area at Hannover, Germany

The previous example compared OTC to current installations in urban environments by taking typical situations into account, which have been (approximately) foreseen by traffic engineers at design time. This design time process is performed using the following classification: Mondays, Fridays (equal to last day before a bank holiday), Saturdays and Sundays have their own characteristics, while Tuesdays to Thursdays can be handled in the same way. Although the traffic profile during the course of the day is not exactly the same at each day, these five categories can be used to handle normal traffic patterns sufficiently. Consequently, traffic engineers define different time-dependent strategies for each of these categories. One crucial aspect not considered in either of these categories is given by non-regular events affecting the normal traffic situation. One of these examples has been introduced in Figure 6.1 when describing the effect of a worldcup football match on the traffic situation at Karlsruhe.

The second scenario is used to evaluate OTC's behaviour in the presence of such non-regular events where the situation is hardly controllable with predefined static solutions developed at design time. An event at *the stadium* takes place generating abnormally high approaching and departing traffic. In terms of OC, this event can be seen as *disturbance*. Consider the current football season in Germany as example to motivate the stadium scenario: November 20, 2010 the local football team Hannover 96 played Hamburger SV in a regular season match (a Saturday, match start at 3:30 p.m.), while one week later on November 27, 2010 Hannover 96 played SC Freiburg again in a regular season match (a Saturday, match start at 3:30 p.m.). The former match is a "derby" against local rivals, which resulted in an attendance of 49.000 people (booked out). In contrast, SC Freiburg seems to be less attractive resulting in a significantly lower attendance of 34.100 people.[3] The former match has attracted about 44 % more people attending the match compared to the Freiburg match, entailing a comparable difference in traffic at the surrounding road network. However, the signal plans of this inner-city road network surrounding the stadium

[3]All attendance figures are taken from `http://www.fussballdaten.de` at December 20, 2010.

area are following the same strategy in both cases – a strategy already developed for such cases and thereby differing from standard signal plans.

Similar to the previous example, the signal plans and the traffic data are modelling a realistic setup.[4] A complete description of the simulation considering traffic data and signalling can be found in Appendix C. The signalling of the traffic network surrounding the stadium is part of the overall traffic strategy for the urban area of Hannover. The intersection controllers are operated by seven different daytime-depending strategies: five according to the initially mentioned classification and two especially designed for stadium events. The latter two control strategies cover the approaching and departing traffic – in general, the former one optimises the network's throughput on the main streets approaching the stadium, while the latter one provides traffic's fast departure of the inner-city region.

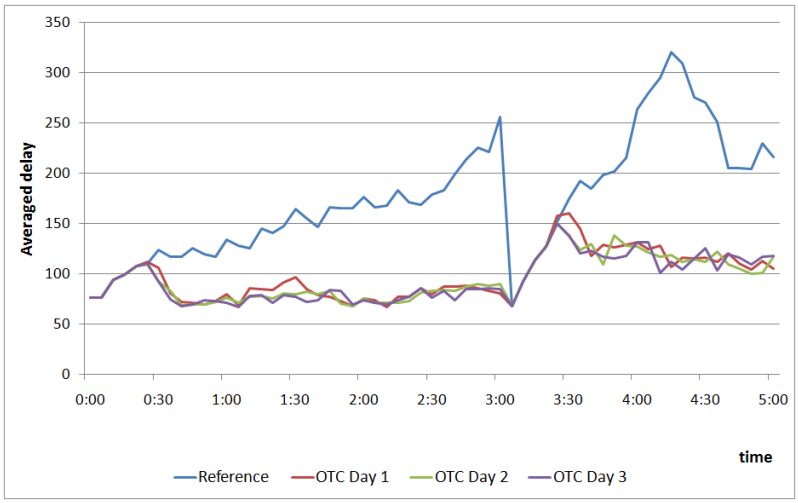


Figure 6.15: Averaged delays per vehicle in the simulation of the stadium area located at Hannover, Germany (lower values are better)

The first part of the simulation-based evaluation covers typical metrics from the domain of traffic engineering again. Intuitively, the goal for control strategies responsible for arriving and departing traffic in the presence of such stadium events is to maximise the throughput of the network. The corresponding local effect at each intersection controller is to minimise the waiting queues and consequently to decrease the occurring averaged delays per vehicle. The following results are obtained as averages from five runs of the particular simulations with different random seeds. Figure 6.15 depicts the achieved results at network-level. The abrupt drop after 3 $hours$ of simulation time is caused by the setup of the simulation. The provided traffic data did not contain figures for the time interval between arrival and departure for the stadium event. Thus, only the arrival and the departure parts are simulated, which leads to

[4] All traffic data and signal strategies have been provided by the Landeshauptstadt Hannover, Fachbereich Tiefbau (Bereich Koordinierung und Verkehr) and the Verkehrsmanagementzentrale Niedersachsen (Region Hannover). The traffic data has been obtained from a census performed on May 09, 2009. The *Aimsun* models have been built in cooperation with *Weinreich* [272].

an abrupt change in the traffic situation. The figure shows four curves for simulations with the same data. The reference solution leads to an averaged delay of 174 s per vehicle. The "OTC Day 1" line represents a simulation of the OTC system starting with an empty rule base. Obviously, the adaptive control strategy of OTC outperforms the reference solution already with limited knowledge. The simulation of the first day resulted in an averaged delay of 97.1 s per vehicle, which corresponds to a decrease of 44.2 % compared to the reference solution. In the following days, the same event is repeated and the OTC system recognises the learned traffic situations. Consequently, an additional benefit is visible. The "OTC Day 2" simulation resulted in a decrease of 45.6 % (delay of 94.7 s per vehicle) and "OTC Day 3" in a decrease of 46.1 % (delay of 93.8 s per vehicle) – both compared to the reference solution.

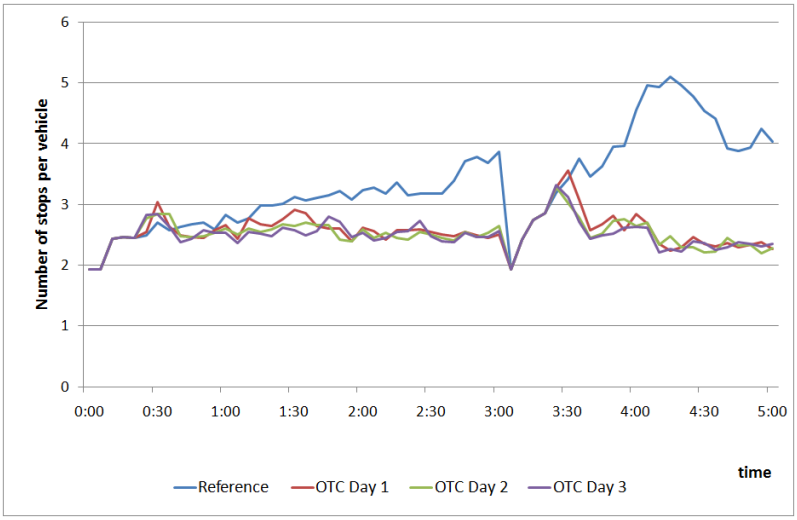


Figure 6.16: Averaged number of stops per vehicle in the simulation of the stadium area located at Hannover, Germany (lower values are better)

Besides the pure delays occurring when crossing the underlying network, the number of stops at red traffic lights is a second major aspect. Figure 6.16 depicts the corresponding results. The simulation of the reference solution results in 3.33 *stops* per vehicle on average, which is reduced by 23.3 % at day 1 to 2.56 *stops* (day 2: 24.5 % to 2.51 *stops*; day 3: 25.0 % to 2.49 *stops*) due to OTC control. Closely connected to delays and stops is the averaged fuel consumption. Figure 6.17 depicts the achieved result. Again, a significant reduction can be observed compared to the reference solution (21.4 $\frac{l}{100\,km}$). During day 1, OTC reduced the fuel consumption by 14.0 % to 18.4 $\frac{l}{100\,km}$, while day 2 (19.6 % to 17.2 $\frac{l}{100\,km}$) and 3 (19.9 % to 17.1 $\frac{l}{100\,km}$) resulted in an even better performance.

Especially compared to the previous Hamburg-based example, the stadium scenario demonstrates the benefit of using OTC. Disturbed and unanticipated situations are handled significantly better than static existing installations. Thus, the scenario supports the

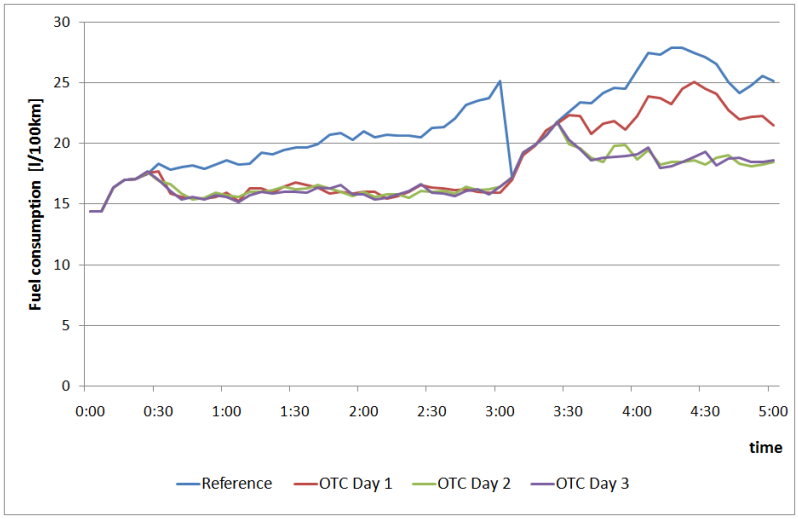


Figure 6.17: Averaged fuel consumption per 100 km in the simulation of the stadium area located at Hannover, Germany (lower values are better)

initially described motivation why adaptive solutions are desirable. The classifier usage and generation analysis showed similar results as in the previous example and is therefore neglected in this context.

6.5.3 A Manhattan-type Test Network

In contrast to the former two scenarios, the third scenario is artificial by means of the investigated traffic network and its configuration. Due to the realistic models and the inherent characteristics of the former two examples, the potential benefit of the coordination mechanisms is not clearly visible or separable enough from the effects caused by the pure OTC control. Thus, the artificial environment provides a possibility to compare the coordination effects achieved by the DPSS and HPSS mechanisms. Since no reference solution exists for such an artificial network, the pure uncoordinated OTC system serves as reference.

The investigated Manhattan-type network consists of six intersections with identical topology (see Figure 6.18). Each road is defined as single-laned road with a length of 250 m and provides an extra side-lane for left-turning traffic in front of signal lights. The setup of the traffic demand has been chosen according to the motivating example as depicted in Figure 6.6. In particular, the expected effect is a sub-optimal coordination when using the DPSS mechanism and an optimal coordination when using the HPSS approach. The corresponding traffic demands are listed in Table 6.1. During the first two simulated hours, the most prominent traffic stream is the stream running from intersection G to intersection H (see Figure 6.18 for labels). Consequently, the DPSS mechanism will establish PSSs for the north-south streams. This is less efficient than creating PSSs for the west-east streams

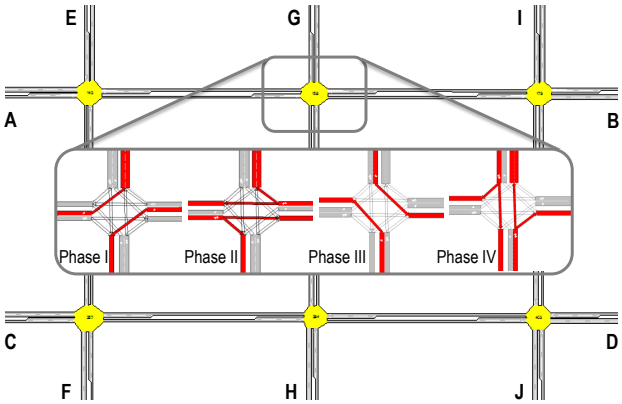


Figure 6.18: Simulation model and signal phases for an artificial Manhattan-type network

O/D pair	1st half		2nd half	
	$\rightarrow$	$\leftarrow$	$\rightarrow$	$\leftarrow$
$A \leftrightarrow B$	450	200	200	450
$C \leftrightarrow D$	450	200	200	450
$E \leftrightarrow F$	200	100	100	200
$G \leftrightarrow H$	575	200	200	575
$I \leftrightarrow J$	200	100	100	200
Others	10	10	10	10
Total	3475		3475	

Table 6.1: Simulated traffic demands (in $\frac{vehicles}{hour}$) for the investigated Manhattan-type network

from A and C to B and D, respectively. The latter coordinations are preferred by the HPSS mechanism using aggregated network-wide data and therefore detecting a higher potential in terms of benefiting vehicles. For the second half of the simulation, the traffic demands change their directions – as a result, the coordination mechanisms have to detect the new demands and adapt the PSSs. The simulation starts with empty rule bases in all cases. The PSSs are checked and potentially updated every ten simulated minutes. The following results are obtained as averages from three runs of the particular simulations with different random seeds.

The first part of the experimental evaluation considers the travel times needed to pass the network. Figure 6.19 depicts the results. The uncoordinated OTC system (running neither the DPSS nor the HPSS mechanism) leads to an averaged travel time of 181.3 s for passing the network. This value has been slightly reduced by both mechanisms. Using the DPSS mechanism in addition to the pure OTC control resulted in a decrease of 3.02 % (175.8 s) and HPSS in a decrease of 3.92 % (174.2 s). The goal of the coordination is to reduce stops in front of red traffic lights – the corresponding values are depicted in Figure 6.20. Considering the figure, a significant reduction of stops caused by both mechanisms can be observed. For

the complete simulation period, an average reduction of 3.9 % for the DPSS and 9.4 % for the HPSS mechanism is obtained. These results reflect the initial expectation – the HPSS mechanism achieves higher reductions in terms of stops, but even the performance of the DPSS mechanism results in a significant improvement in comparison to the uncoordinated system. Due to the abrupt change in the traffic conditions after two hours of simulation time, the scenario does not provide ideal conditions for the coordination mechanisms – the change causes temporarily increased travel times and stops for both approaches. However, the benefit of the two coordination mechanisms is visible.

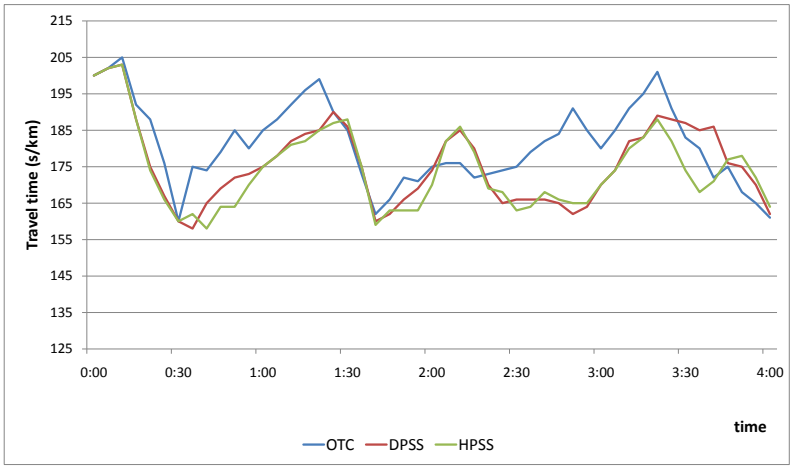


Figure 6.19: Network-wide travel times for the Manhattan-scenario (lower values are better)

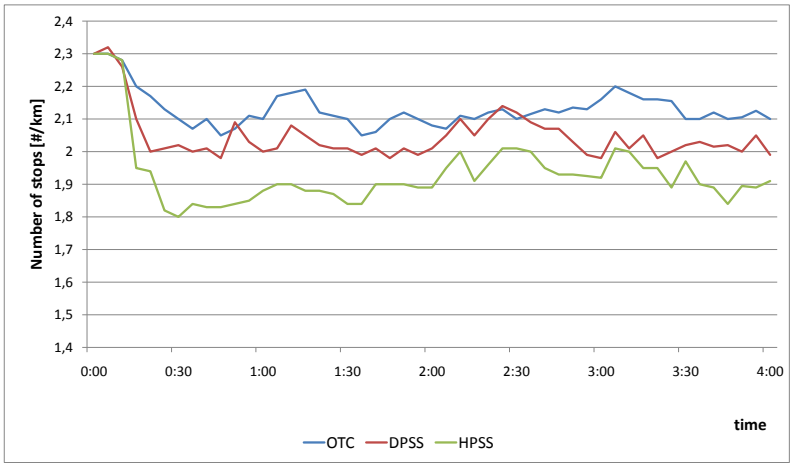


Figure 6.20: Network-wide number of stops for the Manhattan-scenario (lower values are better)

The previous results illustrated the network-wide benefit of establishing PSSs. To illustrate the differences between both approaches and to demonstrate the effect of coordination

at stream-level, two prominent streams have been investigated in detail. The Stream $G \rightarrow H$ is the most heavily used one during the first two simulated hours – consequently, this stream is chosen by the DPSS mechanism to establish a coordination for this period. After the abrupt change, it is no longer coordinated and as a result travel times and stops re-increase to the level of the uncoordinated operation. During the complete simulation period of four hours, a reduction of 11.5 % with respect to travel times and 19.8 % with respect to stops is obtained for this traffic stream by the DPSS mechanism.

As intended by the initial setup, the west-east Stream $A \rightarrow B$ is coordinated by the HPSS approach during the first half of the simulation due to the higher potential benefit. After the change in the traffic situation, the stream is no longer coordinated. For the whole simulated period, the expected effect can be observed. The travel time has been reduced by 11.9 % for all vehicles following the stream from A to B. Even more significant is the averaged number of stops: it has been decreased by 36.9 %. Table 6.2 lists the improvements of both approaches compared to the reference solution.

	DPSS		RM	
	Travel time	Stops	Travel time	Stops
Network	3.0 %	3.9 %	3.9 %	9.3 %
Stream $A \rightarrow B$	−0.6 %	3.1 %	11.9 %	36.9 %
Stream $G \rightarrow H$	11.5 %	19.8 %	3.7 %	4.0 %

Table 6.2: Reduction of travel times and stops compared to uncoordinated operation in the Manhattan-type network (higher values are better)

A major goal of coordinating the control strategies of intersections is to reduce the environmental impact of traffic. To analyse the effects caused by both mechanisms, AIMSUN's environmental models have been used to evaluate the vehicles' fuel consumption and their emission of pollutants. During the complete simulation period, the uncoordinated OTC system serving as reference resulted in an averaged fuel consumption of 14.1 $\frac{l}{100\,km}$. This value has been reduced by the DPSS mechanism by 4.5 % to 13.4 $\frac{l}{100\,km}$. Even better is the performance of the HPSS mechanism, which reduced the fuel consumption by approximately 9.8 % to 12.7 $\frac{l}{100\,km}$. According to the models from [271], the presented reductions for fuel consumption can be directly mapped to e.g. the emission of Carbon Dioxide (CO_2). For a given type of fuel, a vehicle's CO_2 emissions are directly proportional to the quantity of fuel consumed. Consequently, the coordination using DPSS reduced the CO_2 emissions by approximately 3.5 % and using HPSS by approximately 6.6 %. Thus, the coordination using both approaches has a significantly positive effect on the environment.

In summary, both presented traffic-adaptive coordination mechanisms have a beneficial impact on the network's traffic conditions. All relevant values for travel times, number of stops, and emissions have been significantly decreased. The investigated test scenario has been designed to exploit a weakness of the DPSS mechanism – in most other cases, both approaches will establish the same PSS systems. Although the HPSS mechanism

outperformed the DPSS mechanism in the investigated scenario, both approaches provide a significantly better solution than the uncoordinated system.

6.6 Summary for the Organic Traffic Control System

This chapter presented the *Organic Traffic Control* (OTC) system, which is based on the presented framework. OTC has been developed to enable a traffic-responsive adaptation of the control strategies at urban intersections. In addition, the system incorporates decentralised coordination mechanisms for establishing Progressive Signal Systems (PSS), to automatically detect incidents in the underlying road network and for traffic-adaptive routing of traffic participants. The chapter introduced the general control problem and described the necessary customisation of the general framework to match the demands of traffic control. In addition, the current state of the art in traffic control has been presented.

The experimental evaluation introduced three test scenarios:

- a model of an intersection situated at Hamburg, Germany,
- a complete model of the road network surrounding the stadium at Hannover, Germany, and
- an artificial Manhattan-type road network.

The former two scenarios represent real conditions at the modelled installations including the corresponding traffic demands and signal plans. The latter scenario uses artificial values and a regular network configuration in order to investigate the benefit of additional coordination mechanisms. Therefore, typical metrics from the domain of traffic engineering have been used: *travel times*, *delays* occurring at intersections, *Level of Service*, *number of stops*, and *emissions*.

The first scenario at Hamburg showed that the OTC solution outperforms the existing installation taking the actual signal strategy and traffic data from a census into account. Within the simulations, the delays caused by waiting cars in front of red traffic lights have been reduced by 16 % compared to the reference solution. In combination with a 5 % reduction of the number of stops, the negative environmental impact of traffic has been decreased by 5.1 % in terms of fuel consumption. In addition, the evaluation analysed the generation and usage of rules for the on-line learning component of Layer 1. The second scenario investigated an atypical traffic event. Based on real-world traffic data and signallings again, the traffic control strategy of OTC has been compared to a reference solution in case of a *stadium event* taking place. The evaluation showed that such a special situation is handled with even higher benefit as the previous undisturbed example. For instance, the delays have been reduced by 46.1 %.

The third experiment introduced an artificial Manhattan-type network to analyse the impact of the two developed mechanisms to establish PSSs: DPSS and HPSS. Since no

reference solution from the real world is available, an artificial setup has been chosen, which prefers the HPSS approach. In general, the benefit of using both approaches has been demonstrated, since the values of all relevant metrics have been significantly reduced. In summary, the chapter demonstrated the benefit of using OTC and therefore the developed framework in the domain of traffic control.

Since the OTC system is based on joint work with other project partners, the following Table 6.3 intends to distinguish the author's own work from the project's general results.

Content	*Project work*	*Author's work*
Analysis of the state of the art	-	X
General system	X	-
Customisation of architecture	-	X
DPSS	X	-
HPSS	X	-
DVR Routing	X	-
LSR Routing	-	X
AID	-	X
Evaluation	-	X
Hamburg Scenario	X	-
Hannover Scenario	-	X

Table 6.3: Assignment of OTC parts to project work and author's own work

Chapter 7

Organic Network Control

The development of networking during the past decades is characterised by an increasing number of protocols proposed for different applications at all layers of the protocol stack. This is partly due to the ubiquitous availability of networked devices for a wide spectrum of applications, and ranges from classical desktops and servers in wired networks to small handheld and embedded devices in wireless networks, such as mobile phones and sensor nodes. This development is accompanied by the users' requirements of a better convergence of all these networks and applications.

Most protocols offer a large number of parameters that allow for customising them to different usage scenarios, e.g. they allow for changing settings for timeouts, number of nodes to connect with, and retransmission counters. However, these parameters are seldom changed at runtime. Instead, they are mostly investigated and set at design time or – at best – changed manually at runtime. This leads to a rather static configuration even though the situation in the network is constantly changing. These arising dynamics are not only characterised by changes in – for instance – available bandwidth, network topology, and channel quality over time. Additionally, new applications (and protocols respectively) are introduced. As one example, the class of Peer-to-Peer applications is probably one of the most dynamic classes of applications contributing to the changes in the protocol landscape during the past years. In essence, well-established protocols (e.g. for web traffic) have to co-exist with protocols that no one would have thought of some years back in the first place.

Considering the developments in data communication and the provoked upcoming demands, research initiatives predicted new challenges and correspondingly the need of new technologies [273]. As one example, *Hummel et al.* describe their prediction that especially protocols for the Internet will be subject to adaptations, since the increasing mobility caused by handhelds and smartphones leads to new requirements for current protocols [274]. Other

researchers state that completely different basic technologies are needed to tackle the upcoming demands [275]. Since some networks are already reaching their load limits, fast solutions are needed, which are compatible with existing technologies – the parametrisable character of network protocols constitutes a promising application domain for the developed framework. The special properties of network protocols makes them an ideal testcase. Protocols have to cope with long-term trends (increasing data volume), short-term changes (peaks in the transfer loads), or temporary malfunctions (node failures).

This chapter introduces the **Organic Network Control** (ONC) system as one implementation of the thesis' general architecture. The ONC system allows for "wrapping" existing data communication protocols into the general framework, which enables a large degree of adaptivity based on self-organisation in existing networks. Initial research for the ONC system focused on the question, whether there is a significant benefit due to scenario-dependent reconfiguration of the protocol parameters or not. Using the examples of a mobile ad-hoc network (MANet)-based broadcast algorithm (the *R-BCast* protocol by *Kunz* [276]) and a smart-camera protocol (the *ROCAS* protocol by *Hoffmann et al.* [277]), a standardised system to optimise network parameter configurations for given scenarios has been developed [278].

Further on, the concept of adaptive protocol control using ONC has been outlined in 2009 [4] and since applied to different protocols ranging from *reliable broadcast protocols for MANets* [279, 280] over *wireless sensor network* (WSN) protocols [281] to *Peer-to-Peer* (P2P) protocols [282]. A summary of ONC in the context of biologically inspired networking and sensing can be found in [121], a current status description and an outlook on future activities is given in [283].

The chapter is organised as follows. Initially, the problem to be covered by the ONC system is defined. Afterwards, related work in the field of adapting network protocols to dynamic changes in the environmental conditions is discussed. Afterwards, the application of ONC to the previously mentioned protocols is explained by introducing examples for MANet, WSN, and P2P protocols. All three examples are evaluated in simulation-based settings. Using the achieved results, the benefit of the additional ONC-control is demonstrated in comparison to the existing static protocol solutions. Finally, a collaboration mechanism is introduced, which allows for the exchange of knowledge and a dynamic load balancing of Layer 2 tasks. The chapter closes with a summary for the ONC system.

7.1 Problem Description

The performance of a network protocol depends strongly on the configuration of its parameter set. These parameters are highly *protocol-dependent*. Although there is no complete list, some classes of protocols share the same functionalities and therefore similar parameters. For instance, protocols for MANets typically have intervals to send "hello"-messages and a

counter indicating after which number of not-answered "hellos" a neighbouring node is assumed to be outside the sending distance due to mobility. Even more popular are protocols from the Internet, especially the TCP (parameter: delays, timeout, size of receiver window, etc.) or the IP protocol (TimeToLive, etc.). Hence, researchers and engineers developing new protocols or adapting existing ones try to find a standard configuration, which works best on average in all considered situations. The underlying situation space is assumed to be indefinite due to unforeseen interdependencies and mutual influences – thus, only a small fraction of these situations can be considered during the design process. Consequently, the task of ONC is to provide an adaptation mechanism that chooses the best parameter configuration according to the observed environmental conditions and the corresponding node's status.

Not only the parameters differ when considering varying protocols, but also the important characteristics of the observed situations. Consequently, the control problem of ONC cannot be defined straightforward as done for OTC in the previous chapter. Thus, not a single controller – which is a network protocol instance in this case – is investigated in different settings, but a set of different protocols. Correspondingly, varying requirements appear that have to be covered by protocol specific engineering solutions that are discussed in accordance with the particular protocol scenarios.

7.2 Related Work

Adaptive control of data communication networks is a research domain focused from several points of view. The amount of literature covering aspects of the topic is growing larger every year. The following section gives an overview of how network protocols are configured, introduces related approaches, and describes the differences to the ONC system.

7.2.1 Determine Protocol Parameter Configurations

As outlined before, the performance of a network protocol depends highly on the configuration of its parameter set. Thus, a main task for engineers is to configure protocols and find the best settings. This configuration process is usually performed at design time when new protocols are developed or existing ones are adapted. In order to determine the best-fitting set of parameters, a network engineer can try to choose parameters manually and continue using a directed *trial-and-error* approach. Alternatively, he can rely on an automated system since the effort for manual optimisations increases exponentially with the number of parameters and the size of the configuration space.

Due to monetary and safety reasons, the testing of network protocols can hardly be performed in real environments. Consequently, simulation tools are used to model the

reality and analyse the protocol's behaviour. Currently, the most popular tools to simulate data communication networks in research and industry are *NS-2* [130] and *Omnet* [131] – an overview and comparison of current simulation tools is given by *Weingärtner et al.* in [284].

In contrast to manual parameter configurations, ONC chooses and adapts parameter settings automatically at runtime. The process of automatic parameter configuration to fine-tune the settings at design time has been investigated depending on specific protocols, which resulted in scenario-dependent solutions. One approach has been presented by *Montana and Redi* in 2005 [285] – they use an EA to optimise a protocol for military MANets. Another example using an EA is the optimisation of parameter settings as described by *Sözer et al.* [286] for an underwater communication protocol. Thirdly, *Turgut et al.* [287] discuss the usage of a Genetic Algorithm to optimise their MANet-based clustering protocol. They all compare their achieved results to a manual optimisation. All three systems are used as representatives for network engineers developing solutions to automatically fine-tune their protocols at design time – the authors' intention in these cases has been to optimise a specific protocol and not to create a generic system. The following part of this section discusses possibilities to adapt parameter settings automatically at runtime.

7.2.2 Automatic Protocol Adaptation

In recent years, automatically performed on-line adaptation of network protocols has become a highly dynamic field of research. A broader community has been generated in the course of IBM's *Autonomic Computing* (AC) initiative [1] and the subsequent *Autonomic Communications* initiative [107]. Compared to the off-line configuration as described previously, on-line adaptation is a significantly more complex task – due to time and computational restrictions. Besides ONC, different directions of research are known to cope with the problem: *adaptive protocols*, *composition of protocol stacks*, *centralised solutions* to adapt protocol configurations, or *self-parametrising systems*.

Adaptive Protocols

The most obvious way of adapting network protocols to changing environments is to consider adaptivity aspects within the protocol logic. Thus, several examples can be found in literature ranging from the *Media Access Control* (MAC) up to the application layer. As one MAC-based example serves the work presented by *van Dam and Langendoen* [288]. They focus on contention-based media access. To handle load variations in time and location, an adaptive duty cycle is used by ending the active part of it dynamically. Similar examples for the MAC layer are e.g. given by *Huang et al.* [289], or *Farago et al.* [290, 291].

One layer upon the MAC layer, *Goyal et al.* presented an adaptive network layer protocol

that aims at minimising buffer requirements within the network without losing packets. Simultaneously, it tries to minimise end-to-end delays and jitter of frames [292]. The authors introduced a receiver-oriented, adaptive, and credit-based flow control algorithm capable of adapting the settings of the protocol according to observed attributes. On the same layer, *Whiteson and Stone* introduced an on-line learning mechanism to increase the performance of a routing protocol [293]. Based on the Q-routing techniques presented in [294], they learn the best routes by receiving immediate answers from the next hop. Enlarging the focus, *congestion avoidance* in TCP [295] and *collision detection* for Ethernet-protocols [296] can also be seen as adaptive protocols, since they react on observed stimuli.

All these protocols provide specific solutions – they are designed to enable adaptive behaviour just for one task, mainly situated at the MAC layer. These are reactive approaches and rely on pre-estimated configurations and actions. Thus, they cannot be applied to other protocols, layers, or tasks.

Protocol Composition

Since developing protocols with adaptive logic for all purposes is infeasible and the set of systems requiring adaptive communication services (e.g. services adapting their behaviour according to changes in the environment) is increasing simultaneously, a research field called *protocol stack composition* emerged, covering the upcoming tasks by exchanging protocols and stacks dynamically [297]. In contrast to ONC, which keeps the existing and currently used techniques and optimises their behaviour, a re-combination of protocols is performed. Although the goal deviates from the ONC approach (i.e. the protocol stack exchange has impact on all involved systems and can hardly be done locally), the approach enables adaptivity aspects at protocol-level.

As a first basic aspect, protocol composition and execution frameworks aim at simplifying the usage, design, and configuration of data communication protocols. The more complicated a protocol is, the harder it can be exchanged against another comparable or (depending on the situation) more suitable one. This leads to the insight that protocols should be designed in a more modular way. Hence, the goal of protocol composition is to provide the opportunity of composing protocol stacks according to the application needs. At runtime, the framework covers the necessary monitoring and data exchange tasks to provide knowledge for deciding on the best available protocol composition. The most important frameworks covering these tasks are *Appia* [298], *Cactus* [299], *Consul* [300], *Ensemble* [301], *Horus* [302], and the system presented by *Mena et al.* [303]. Besides the locality aspect, some characteristics of the approaches separate them from the requirements of the ONC framework. For instance, the protocols and their configurations have to be known in advance and further automated extensions with new behavioural repertoires are not intended.

Furthermore, *Schöler and Müller-Schloer* presented an adaptive monitoring architecture for protocol stack configuration which already focuses on the techniques used within the

ONC system [304, 305]. In particular, it makes use of OC's *Observer/Controller pattern* (see Chapter 2.2). Similar to ONC, the authors' approach relies on self-optimisation based on learning at runtime – thus, they describe the application of a *Fuzzy Learning Classifier System* [168] to cover the protocol exchange decisions and to learn from the feedback prepared by the observer. But, unlike the ONC framework, the system is again built without offering the opportunity of handling different protocols and extending the set of possible solutions autonomously and on demand.

Centralised Systems for Protocol Adaptation

The previously discussed approaches have drawbacks, especially in the limited repertoire of possible actions. As mentioned for protocol stack composition, researchers focused on finding network-wide solutions to adapt protocols to current demands. Besides choosing matching protocols, another option is to change their configuration. First attempts proposed centralised systems capable of adapting the protocol's configuration dynamically at runtime. In 2001, *Sudame and Badrinath* [306] presented a first example. They introduced a first TCP- and UDP-based study and defined the need of dynamic adaptation, but detailed examination and a demonstration of the re-usability for other protocols have not been addressed.

In the same year, *Ye et al.* presented their system, which is based on a centralised adaptive random search algorithm that tries to combine the stochastic advantages of pure random search algorithms with threshold-based knowledge [307, 308]. This search heuristic is used to determine the best-fitting parameter settings for the entire network – all contained nodes are configured *centrally* with the same settings. Recently, they extended their approach towards a back-end support tool for large-scale parameter configuration that is based on efficient parameter state space search techniques and on-line simulation [309].

Another related approach is introduced by *Georganopoulos and Lewis* in [310]. They describe a dynamic optimisation framework for the reconfiguration of network protocols at all layers of the protocol stack. The approach is based on previous work done for adapting configurations of mobile terminals in a TCP/IP setting [311, 312]. Their system tries to optimise its performance according to given goals by adjusting or replacing different entities (applications, protocols, etc.). All these approaches rely on a centralised element and adapt protocol settings for all affected nodes in the whole network. To allow for such a division of work between a central server and the particular nodes, problems like *bandwidth usage*, *single point of failure*, or *local knowledge accessible from server-side* have to be covered. Besides these three systems, further approaches (especially protocol-specific ones) are known, but the presented ones are those most similar to the scope of ONC.

Self-parametrising Systems

Since all solutions presented before have several drawbacks due to the need of a central element, research focused on developing self-organising approaches – like ONC. As one example, *Su et al.* [313] presented their approach for a mobility-adaptive self-parametrisation of different unicast and multicast routing protocols in the MANet-domain. The approach assumes that each node can determine its position by a positioning system (e.g. GPS). By periodic position measurements, a node can additionally determine its velocity and its moving direction. This information is included in the control messages of the routing protocols. Thereby, mobile nodes get informed about the position, speed, and moving direction of other nodes. Based on this information, link expiration times and route expiration times are computed as a basis for configuring routing parameters like update intervals, or to select stable routes.

In contrast, *Ahn et al.* [314] do not depend on the existence of a positioning system. Instead, they use the observed number of changes in the single-hop neighbourhood of the nodes as mobility metric. The nodes exchange lists of new and lost neighbours in the last interval and calculate a metric describing the dynamics in the neighbourhood. Following this metric, a dynamic adaptation of the protocol is achieved. In comparison to the ONC approach, the authors rely on a protocol extension that is responsible for updating and exchanging these lists. Furthermore, the approach is only feasible for the MANet-domain.

Similar to *Ahn et al.* [314], *Boleng* [315] determines the mobility by analysing the observed neighbourhood locally. Considering the average link duration as mobility metric, the author uses the received values to control the data forwarding of his *Adaptive Location Aided Routing from Mines* (ALARM) routing protocol (an extension of [316]). If the average link duration exceeds a given threshold, low mobility is assumed and the packets are forwarded by *normal* routing. In contrast, high mobility is assumed and the data packets are flooded if the average link duration falls below this threshold. Again, this approach is a specific solution just for one protocol.

The most recent development has been presented by *Stanze et al.* in 2006 [317]. They describe a system for mobility-adaptive self-parametrisation of a routing protocol in MANets. Therefore, they measure the mobility by using their *MANET Relative Velocity Indicator* (MARVIN) protocol. Based on continuously measuring MARVIN and the corresponding observed situation, the authors adapt the parameters of their routing protocol. All approaches named before are restricted to specific routing protocols in MANets. They are application-specific and cannot be transferred to enable self-parametrisation of protocols from other domains, since they focus on determining the mobility aspect in MANets and use predefined actions for observed situations.

7.3 Application of the Generic Architecture

Since the ONC system is designed to control several different network protocols, a general adaptation to a specific application is not possible. Thus, a protocol-specific solution is needed, which is described in the remainder of this section. Initially, a broadcast protocol for MANets is investigated, which is situated at lower layers of the OSI protocol stack [318]. Afterwards, the application layer is focused in the context of wireless sensor networks and Peer-to-Peer networks. On the one hand, these three scenarios have been chosen to demonstrate the generic character of the ONC system. On the other hand, different aspects of ONC can be explained using the particular characteristics of the protocols.

7.3.1 Broadcast Algorithms in Mobile Ad-hoc Networks

A demanding challenge for the ONC system is investigated in the context of the first scenario: the control and adaptation of MANet protocols. The possible movements of nodes lead to continuously changing situations: neighbours are getting out of reach or joining the sending distance. Hence, highly dynamic environments and vast situation spaces occur making MANets a promising application area. An important mechanism to distribute messages in MANets is broadcasting – these algorithms are used as a basic technique in further protocols on higher layers of the protocol stack (e.g. routing).

In order to apply ONC to MANet-based broadcast protocols, the exemplary R-BCast protocol as introduced by Kunz [276] has been selected. In the following section, the protocol and its parameters are explained. Afterwards, the adjustments of the general framework as introduced in Chapter 3 are discussed. Therefore, the basic tasks as formulated for wrapping a new system into the framework are fulfilled. Based on an experimental simulation environment, the achieved results of the evaluation are analysed and explained. The scenario as discussed in this section has been initially published in [279]; an extended approach can be found in [280].

Protocol

The *Reliable Broadcast Protocol* (R-BCast) as introduced by Kunz in [276, 319] has been chosen, since it is representative for the domain. The goal of the protocol's development has been to achieve reliability when delivering messages in ad-hoc networks and to increase the packet delivery ratio compared to other protocols. Therefore, the author designed techniques like equipping the nodes with extra buffers to retain messages in temporary storage. These round-robin-based buffers are used to store the last p received unique packets. In contrast to other protocols, the R-BCast protocol has significantly more variable parameters. Consequently, the task of controlling the protocol is more complex, but – in turn – it also offers a higher potential benefit to be gained from a dynamic adaptation.

Parameter	*Standard configuration*
Delay	0.1 s
AllowedHelloLoss	3 *messages*
HelloInterval	2.0 s
δHelloInterval	0.5 s
Packet count	30 *messages*
Minimum difference	0.7 s
NACK timeout	0.2 s
NACK retries	3 *retries*

Table 7.1: Variable parameters of the R-BCast protocol

Parameters

The R-BCast protocol has several variable parameters, a detailed discussion of the purpose and the functionality of each of these parameters is given in [276]. Table 7.1 lists them with their particular standard configuration. In order to understand what the purpose of each parameter is and what effect an adaptation can have, they are briefly discussed in the following.

The *delay* variable is used to define the maximum deceleration time between receiving and forwarding a message. Due to the mobility in MANets, many protocols are built on exchanging "hello"-messages to keep track of their neighbours' availability. Therefore, the R-BCast protocol has an *AllowedHelloLoss* variable defining the maximum of "hello"-messages, which may be lost until a neighbouring node is assumed to be out of transmission range. Furthermore, the *HelloInterval* defines the duration between sending "hello"-messages, and the δ*-HelloInterval* randomises this interval to avoid collisions. Besides managing a list of available neighbours, the protocol has a mechanism to hold broadcast messages in temporary storage – if a new node joins the sending distance, lists of the last n broadcasts (and not-acknowledged messages) are exchanged and missed messages are updated. Therefore, the *Packet count* parameter defines the number of messages in the temporary storage – the *Minimum Difference* defines the lowest border of the duration between two not-acknowledged messages to be handled as different broadcasts. The last two variable parameters are also part of this *not-acknowledged* (NACK) mechanism. These NACK messages are used to get missed broadcast messages. If a node has a broadcast message from Sender $S1$ with $ID4$ in its queue and receives a new one with $ID6$, it asks for the one with $ID5$ using a NACK message. In the process, *NACK timeout* defines how long the node waits for an answer for the NACK message until it repeats the process. *NACK retries* specifies how often the process is repeated.

Customisation of ONC

The application of ONC to a new protocols requires the completion of the five basic customisation tasks as introduced for the general system in Chapter 3. Since the approach aims at providing a re-usable framework, an out-of-the-box solution is hardly possible. Thus, the framework provides the general mechanism and functionality, which has to be extended at only a few points to be able to control a new system. The customisation according to the five steps is explained as follows.

(1) Situation Description The first task when applying ONC to the control of a new protocol is to define what is relevant for the adaptation process and has influence on the protocol's performance – the *situation* of the SuOC. Considering MANet protocols, the most important dynamic factor is the distribution of other nodes within sending and sensing range. The goal for reliable broadcast algorithms is to deliver messages to all receivers. This has to be reached with minimal effort (in terms of forwarding messages), since forwarding increases the load of the network. The more nodes are within sending distance, the higher is the probability that forwarding the message is not required, since the neighbours have already received this message by another forward.

Assuming that nodes are able to determine the current positions of their neighbours within the sensing range relative to their own position (e.g. based on GPS, see [320]), the situation of the nodes' environment can be described. But when considering exact positions of neighbours based on GPS coordinates, current and previous situations will hardly be identical. In addition, the learning component relies on a classification of situations to keep the set of possible rules manageable. As a result of these considerations, a sector-based approach as depicted in Figure 7.1 has been developed [279], which discretises the search space. For each of the sectors depicted in the figure, the situation description stores the number of neighbouring nodes within the covered area.

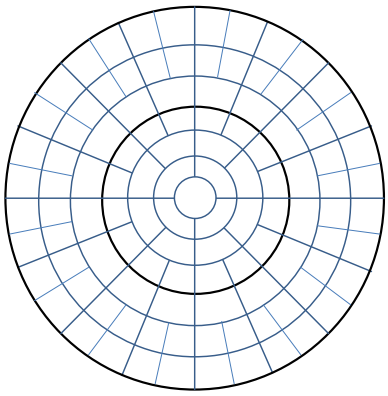


Figure 7.1: Environment representation for MANet protocols

The radius of the outer circle is equal to the sensing distance (*sensDist*) of the node,

as this is the most remote point where messages of this node can interfere with other ones. Typically, the transmission range for Wireless-LAN-based MANets is about $250\,m$ and the sensing distance is twice the sending distance ($500\,m$). The radii of the inner circles have been chosen empirically. Since nodes within the first circle are really close ($50\,m$), their exact positions have only minor influence on the best parameter configuration – probably all neighbouring nodes within this circle receive the same messages. The second circle ($125\,m$) has been partitioned into 4 sectors, the third circle ($200\,m$) into 8 sectors, and the forth circle ($250\,m$, maximum transmission range) into 16 sectors. This corresponds to a trade-off between a classification of the nodes' distribution and keeping the information where the nodes are. The next two circles ($375\,m$ and $500\,m$) are representing the area within sensing range – both circles are divided into 32 sectors each. Besides storing the number of neighbours for each sector, the situation description contains the *node's speed* (if available) and its *direction of movement*, since it has high influence on the best parameter set (e.g. moving towards/away from a set of nodes may influence the best delay value).

(2) Similarity Metric Based on the aforementioned description, a metric to quantify the similarity of two situations is needed, since the learning component of Layer 1 is only allowed to choose from the set of "nearby" rules. Therefore, the distance function $\delta(A, B)$ is defined as follows. The more similar two situation descriptions A and B are, the lower is the distance value determined by the δ-function. Before applying this distance function, the possible influences of rotation and reflection are deducted. Afterwards, the formula for the distance can be defined with $r \in RADII$ and $s \in SECTORS$ as follows:

$$\delta(A, B) = \sum_{r} \sum_{s} (A_{r,s} - B_{r,s})^2 / r.distance \tag{7.1}$$

The function $r.distance$ defines the radius size as introduced before ($50\,m$, $125\,m$, ...). $A_{r,s}$ refers to the number of neighbours within the sector s of radius r for the situation description A. Consequently, the importance of a neighbour decreases with increasing distance.

(3) Learning Feedback Besides the situation description and a distance measurement between situations, the learning component needs a learning feedback to draw conclusions from its past actions. Several metrics have been proposed for MANet protocols – in the context of broadcast algorithms, the most prominent ones are *Packet Delivery Ratio* and *Packet Latency*. Although both cannot be measured locally at each node (they are global figures in the context of Definition 1), they give a hint on what the overall goal is – the system shall reduce the number of forwarded broadcasts and assure the delivery of the broadcast to each node at the same time. To achieve this, the following fitness function ($Fit(x)$) has been developed:

$$Fit(x) = \frac{\#RecMess}{\#FwMess} \tag{7.2}$$

In Equation 7.2, x represents the currently observed network protocol instance. Since a new parameter set has to be applied for a minimum duration to show its performance, *evaluation cycles* are used defining discrete time slots – the control loop consisting of Layers 0 and 1 is performed once every evaluation cycle. The duration of these cycles depends on how dynamic an environment is. The faster the environment changes, the shorter is the cycle (and the more often is the SuOC adapted). Thus, the formula above takes all messages sent and received within the last cycle into account. It divides the sum of all received messages ($\#RecMess$) by the sum of all forwarded messages ($\#FwMess$). As a result, high effort (unnecessary forwards) and low delivery rates (not successful broadcasts) are penalised.

(4) Configuration Space Furthermore, the variable parameters to be controlled by ONC have to be defined as the fourth task of the adaptation process. Here, all variable parameters as introduced by *Kunz* are covered by ONC (see Table 7.1). ONC aims at providing a black box solution and does not interfere with the SuOC's logic – thus, only those parameters accessible by the configuration interface are considered.

(5) Simulation Model Finally, Layer 2 has to build adequate simulation scenarios from the information obtained by Layer 1. Therefore, the network simulation tool *NS-2* [130] is used to perform the simulation tasks. *Kunz* provides a standard implementation of the R-BCast protocol for NS-2, which has been applied to ONC. The situation description is converted into a simulation scenario for NS-2 by creating a randomised instance of the sector-model. The resulting distribution of the neighbouring nodes stays at the given position during the simulation time and the investigated node moves according to the observed speed and movement direction. During the simulations, BCast messages are introduced from outside the simulated network. The fitness calculation is again based on the $Fit(x)$ function as discussed before. The simulation duration depends on the desired effects: in MANets, the movements of the agents are the most important factor. Assuming a maximum pedestrian speed of 6 $\frac{km}{h}$, a node moves about 100 m in 60 s. This distance is a good trade-off between keeping the simulation duration low and providing a basis for effects like closing the gap between two nodes by movements. In general, the simulation duration depends on the particular scenario and has to be determined when setting up the simulation model.

Experimental Setup

The experimental setup consists of two parts: the *simulation environment* and the particular *scenarios*. The simulation environment is implemented in JAVA using the Multi-Agent Simulation Toolkit *MASON* [128]. Within this simulator, the R-BCast protocol has been

implemented according to the NS-2 based solution provided by the author of the protocol. Within the simulated area of 1000 x 1000 meters, ten agents are created at random positions and move according to a random-waypoint-model (see e.g. [321]). From these agents, only one is equipped with the ONC system – the other nine agents perform the standard protocol configuration. The sampling rate of ONC's Layer 1 is set to 1 s. The simulation relies on pseudo-randomised movements by taking *seeds* into account – which makes them repeatable and comparable to the usage of the protocol's standard configuration. The investigated simulation period covers seven hours.

The analysis of the ONC system in this setting investigates several different aspects. First of all, the question has to be answered, whether there is a benefit due to additional ONC-control or not. A special focus in this context is set on the development of ONC-control over time. The system starts with a completely empty rule-base and has to use its Layer 2 component to learn new rules. Based on this start from scratch, the performance has to improve with each simulated day. To quantify this desired effect, the following figures have been recorded during the simulation: 1) the number of successfully delivered broadcasts, 2) the averaged latencies of this delivery process, 3) the number of network-wide messages sent, 4) the number of network-wide messages received, and 5) the rule-generation and SuOC-adaptation effort performed by Layer 1 of the controlled node.

The second part of the evaluation focuses on the required effort for the achieved adaptability caused by ONC. Therefore, the developments in terms of a) the rule-generation part and b) the population size are determined.

Results of the Evaluation

The first part of the evaluation analyses the general impact of the additional ONC control in this scenario. ONC's goal contains two aspects: assure the delivery of broadcasts and decrease the overhead needed to achieve this delivery. Since deliveries of broadcasts can only be considered at network-level, this analysis demonstrates the results from a more global perspective. Figure 7.2 illustrates the delivery ratio of broadcast messages. Therefore, the number of received distinct broadcasts has been divided by the number of sent broadcasts. The figure illustrates the desired effect. The reference solution reported a delivery ratio of 4.63 on average, which is clearly improved by the ONC-control. Day 1 resulted in an increase of 7.38 % (delivery ratio of 4.99), day 2 in an increase of 6.93 % (delivery ratio of 4.97), and day 3 in an increase of 6.42 % (delivery ratio of 4.94). Although the results decrease slightly for the three ONC days (which can only be explained by statistical effects), the increase of the delivery ratio is significant. The second aspect in this context is the latency needed to achieve this delivery ratio. The latency values have been determined as the averaged time to deliver a unique broadcast to a receiver. All four values (reference and ONC days 1 to 3) are within a similar range – the maximum deviation between two values is 0.95 %. Thus, the impact of ONC on the latencies can be neglected.

The second part of Layer 1's objective function aims at minimising the overhead needed to deliver the broadcasts successfully. In this context, *overhead* is defined as all non-broadcast messages – in particular, this includes *NACK* messages, "hello"-messages, and re-transmissions. Figure 7.3 depicts the results for all four simulations. The reference solution (all nodes performing R-BCast's standard configuration) resulted in an average number of 9,109.9 overhead messages. This value has been decreased in case of an activated ONC for one agent. Day 1 resulted in 8,549.3 overhead messages (decrease of 6.15 %), day 2 resulted in 8,757.0 overhead messages (decrease of 3.87 %), and day 3 resulted in 8,647.0 overhead messages (decrease of 5.08 %). This means, that the capacity of the network has been increased between 3.87 and up to 6.15 % due to ONC control – which is a significant improvement. Considering the figure, one can observe that the graph for the reference solution is above the ONC lines, except for the fifth simulation hour. In this case, the standard configuration of the protocol leads to better results than the ONC-controlled version. But this effect decreases with longer learning duration. The third day leads to nearly the same results as the reference solution in these situations.

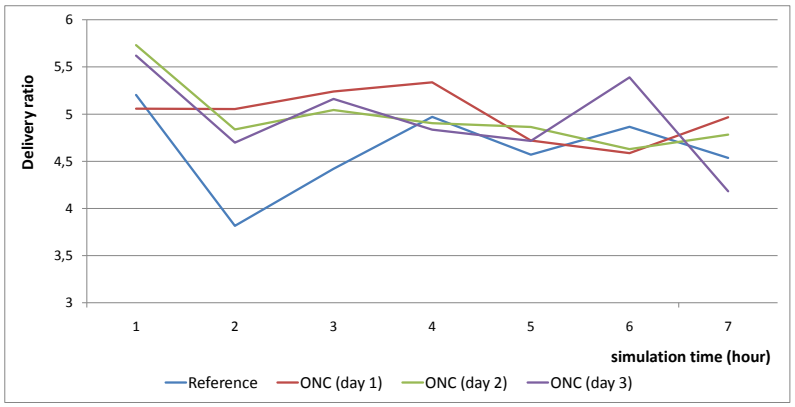


Figure 7.2: Delivery ratio of broadcast messages (higher values are better)

As mentioned for the scenario's setup, only one node is equipped with the ONC system. The advantage of this limitation is that ONC's behaviour can be considered without interferences between mutual learning and adaptation effects – it is traceable which node has been responsible for the changes in comparison to the reference solution. Thus, the following part of the evaluation considers the local results of the *adaptive* node equipped with ONC. Due to the local focus, network-wide figures like latencies and delivery ratios are not measurable. Therefore, the analysis covers the same figures as observed for the on-line learning component when deciding about necessary adaptations. It is founded on measuring the received and sent messages by taking three ONC days and the reference solution into account. As reference, the node's ONC system has been deactivated and the protocol's standard configuration has been performed. Figure 7.4 depicts the results for the received messages and Figure 7.5 depicts the results for the sent messages.

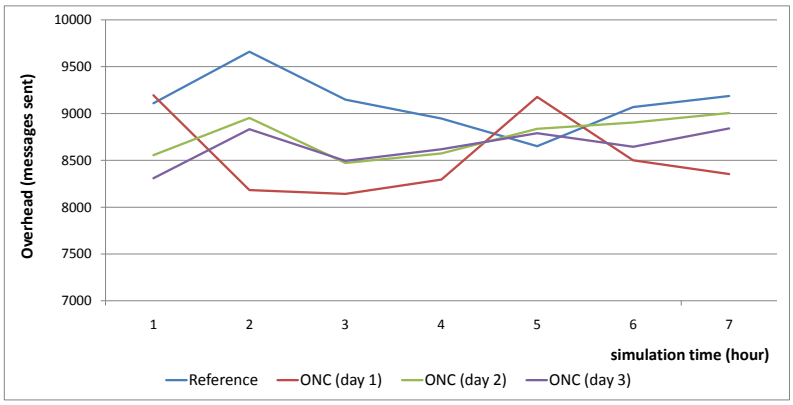


Figure 7.3: Overhead caused by the broadcast protocol (lower values are better)

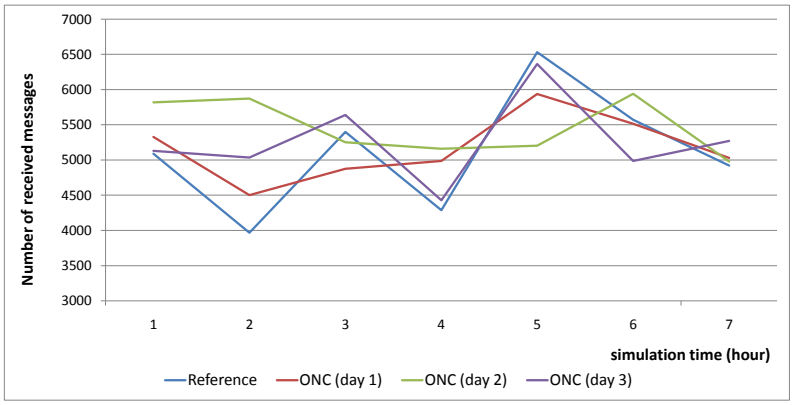


Figure 7.4: Number of messages received by the node under ONC-control (higher values are better)

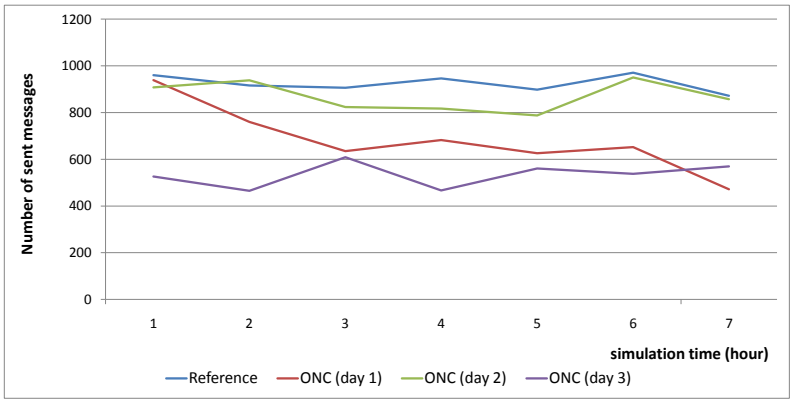


Figure 7.5: Number of messages sent by the node under ONC-control (lower values are better)

As illustrated by Figure 7.4, the reference solution resulted in an averaged number of 5,109.3 received messages. This has been increased by ONC to 5,167.7 (corresponds to an increase of 1.13 %) at day 1, to 5,460.3 (corresponds to an increase of 6.43 %) at day 2, and to 5,265.3 (corresponds to an increase of 2.96 %) at day 3. Considering the averaged results and the different graphs for the four simulations, all of them behave comparably. This observation conforms with the expected effect that a parameter adaptation will probably have a higher impact on sending messages than on receiving them – in case of considering the node's local view.

Hence, the decision about sending messages is in the node's area of responsibility. Consequently, the figure for the *sent* messages is more important for the local view. Figure 7.5 depicts the results. The reference solution sent and forwarded 924.14 messages. This has been decreased drastically by ONC. During day 1, only 680.85 messages have been sent (decrease of 26.33 %), while day 2 resulted in a slightly worse performance by sending 868.85 messages (decrease of 5.98 %). The final day 3 of the simulation showed the best performance with only 533.71 messages sent (decrease of 42.25 % compared to the reference). Thus, the aspect of minimising the sending effort of Layer 1's target function has been achieved successfully – this reduction of the sending effort is caused by reducing "overhead" messages only.

Effort caused by ONC The second major aspect of analysing ONC-control in the scenario is to determine the required *effort* caused to achieve this behaviour. This aspect is concerned with the rule-generation and the SuOC-adaptation behaviour of ONC. Therefore, the following part investigates: a) the development of the rule base over time, b) the closely-connected number of Layer 2 optimisations during the simulation period, and c) the adaptation-demand realised by ONC.

The former two aspects (a and b) are covered by Figure 7.6. The y-axis depicts the number of classifiers, while the x-axis depicts the simulation time in hours. In order to visualise the development through all simulated runs, the x-axis has been defined on an hour-basis showing all three consecutive runs. The figure lists the Layer 2 runs (left bars) and the corresponding population size (right bars) at the end of the particular simulation hour. Since the population contains "covering"-based rules in addition to the Layer 2-generated ones, the population size is not equal to the aggregated number of Layer 2 runs. As depicted in the figure, the population size increases drastically at the begin of the simulation. Starting from hour 14 on, the convergence process begins. Further simulations showed that the number of new rules decreases continuously and the size of the rule base converges to about 3,000 rules. At the end of ONC's day 3, the population contains 2,570 rules that have to be compared to the input stimulus. Due to the convergence, the effort stays manageable. This effect is supported by the observation regarding the utilisation of Layer 2. Obviously, the first day demanded the highest number of Layer 2 runs. The corresponding effort decreases afterwards. In total, 2,170 rule-generations have been performed during the simulated period

(day 1: 1,066 rules; day 2: 925 rules; day 3: 139 rules). Especially by considering the number of Layer 2 runs for all three days, the decreasing behaviour is visible.

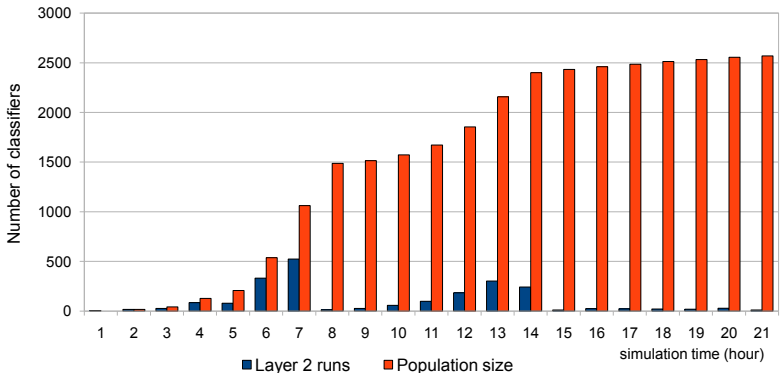


Figure 7.6: Development of the population size at Layer 1 and the performed Layer 2 tasks during the simulation period

Finally, the effort to achieve the ONC-based results has been analysed. The sampling interval has been chosen as 1 s, which means that the ONC-controlled agent had the opportunity to adapt its SuOC 3,600 times per simulated hour. Analysis of the data showed that the Layer 1 component constantly took advantage of this possibility to about 90 % each hour. Thus, every 10th chance to adapt the SuOC has not been used. In general, the number of adaptations corresponds to the speed at which the neighbourhood changes (and consequently to the nodes' movement speeds). Choosing lower movement speeds would decrease the number of adaptations significantly. But as choosing an appropriate rule depends on a linear search of the rule base and the size of the rule base converges to about 3,000 rules, the effort is manageable. Summarisingly, an application of ONC to the control of a MANet-based broadcast algorithm has been successful. The overhead caused by retransmissions and "hello"-messages has been significantly decreased, while the delivery of broadcasts has been improved.

7.3.2 Mode-selection Protocols in Wireless Sensor Networks

The first scenario has been used to investigate the general functionality of ONC and the applicability to MANets. In contrast, the second scenario focuses on further aspects of applying ONC to data communication protocols. So far, it has been assumed that the particular entities performing the protocol are equipped with sufficient resources to cover all aspects of the proposed system. This assumption is not necessarily fulfilled for each protocol or class of protocols that can be controlled by ONC. For instance, research in *wireless sensor networks* (WSN, see [322]) has to cope with limited resources in terms of computational power, storage, and available energy. Consequently, effort has been spent on providing

highly efficient algorithms and communication. Considering the basic design as depicted in Figure 3.2, simulation-based optimisations as performed by Layer 2 are contradicted for the application to wireless sensor nodes. Nevertheless, there is a potential benefit of applying ONC to such restrictive scenarios. Therefore, the scenario is used to investigate cases where a trade-off between *adaptivity* and *resource-usage* is needed. The work presented in the following section has been published initially in [281] and is based on the implementation as presented by *Zgeras* in [323].

As mentioned before, one major problem in WSNs is the efficient usage of resources. Hence, the following scenario has its major focus on the question of how ONC can be adapted to cope with these restrictive circumstances. To address reasonable energy consumption in general, *mode-selection protocols* (see e.g. [322]) have been developed. *Mode-selection* means to determine whether a node can be switched to standby mode for the next cycle or not. By temporarily deactivating nodes, redundancy is reduced when covering the area with the WSN. Consequently, this leads to an enlarged operation time of once-supplied WSNs due to saving energy. Considering the protocol stack, this protocol class can be assigned to the application layer.

This section is organised as follows. Initially, the exemplarily chosen mode-selection protocol is introduced, followed by discussing the variable parameters to be controlled by ONC and their influence on the protocol's performance. Afterwards, the necessary customisation of ONC and the corresponding adaptation of the basic framework are explained. The approach is analysed using a simulation-based evaluation. The setup and the achieved results are presented in the last part of this section.

Mode Selection Protocol

The exemplary mode-selection protocol *Adaptive Distributed Resource Allocation Scheme* (ADRA) as presented by *Lim et al.* [324, 325] has been chosen to demonstrate the benefit of ONC-control under hard restrictions. The ADRA protocol has a relative simple logic and offers just a few parameters to be adapted when deciding whether a node will be switched to standby mode for the next cycle or not. Its mode of operation makes it a representative for the whole domain, and its relative simplicity provides a possibility to analyse the behaviour of ONC. Furthermore, focusing on just a few parameters allows for a processing on sensor nodes, although the investigation presented here is performed using simulation.

The ADRA algorithm distinguishes between three consecutive phases: 1) initialisation, 2) processing, and 3) decision. Following the protocol, each node is able to decide autonomously at the end of the third phase whether it will be in standby mode for the upcoming cycle or not. During the first phase, the node queries the neighbours' mode status, gets information about detected events (if any), updates its local variables (e.g. battery life), and sends information on detected targets to its neighbours. Afterwards, the information is processed in phase 2. The node receives the queried information on events, compares

Parameter	*Standard configuration*
$LocPrio$	0.4
$CovPrio$	0.5
$BatPrio$	0.5
$Thsd$	10.0

Table 7.2: Variable protocol parameters of the ADRA-scheme

its own information about events with the received ones, computes its mode-scheme, and sends this to its neighbours. Finally, the decision about the mode-selection is done in phase 3 by receiving the neighbours' planned schemes, resolving the own plan according to the neighbours' data, and executing the plan (e.g. select standby or active mode).

Parameters

Following ADRA's three-phased algorithm, the mode-selection is determined by calculating a so-called *potential* value (*pot*). The process starts with initialising *pot* for the next cycle. Afterwards, the three *priority* values for influencing aspects are considered: *localisation*, *coverage*, and *battery*. The *localisation priority (LocPrio)* is used, if the node has to locate an event (e.g. triangulation using different nearby nodes). In contrast, the *coverage priority (CovPrio)* is needed to ensure a complete sensing of the environment (the less nodes in the neighbourhood, the higher is $CovPrio$). Additionally, the *battery priority (BatPrio)* covers the aspect of different energy levels among neighbouring nodes. Finally, a *threshold (Thsd)* is needed defining whether the node will be in active or standby mode within the next cycle. For the standard ADRA protocol, all four values ($LocPrio$, $CovPrio$, $BatPrio$, and $Thsd$) are predefined and constant. In this scenario, they will be subject to ONC control and therefore altered at runtime. Table 7.2 lists all parameters and their standard configuration. Since *Lim et al.* do not discuss the settings of the parameters, this configuration is the result of a manual simulation-based optimisation process.

Customisation of ONC

The adaptation of the framework is done by following the five basic tasks as introduced in Chapter 3.Due to the resource questions, the definition of the simulation model is accompanied by a discussion of task distributions for the particular layers of the architecture.

(1) Situation Description Initially, a description covering the current situation of the sensor node and its environment is needed. For WSNs, this environment is mainly characterised by the distribution of neighbours within the node's sending and sensing range in combination with their energy levels. Similar to the MANet-based scenario before, an abstract classification of the neighbourhood is needed. Due to the restricted resources for

wireless sensor nodes, a detailed representation as introduced for the MANet-based scenario is not feasible. Hence, a further abstraction is chosen, which aggregates more situations into one class of situations. In addition, an expensive calculation based on distinguishing between different radii and sectors is not feasible. The result of these considerations is a modified sector-based approach with a simplified partitioning according to the direction of the particular neighbour (Figure 7.7). The figure describes the classification of an observed neighbourhood and its representation as done by each node. For each of these sectors (see Sector 1 to Sector 8 as depicted in Figure 7.7), four different values are stored in the situation description: the number of neighbours in active or passive (standby) mode with more/equal or less energy than the current node. In total, 32 values are stored representing a counter of nodes (four types of nodes, eight sectors – 32 different entries).

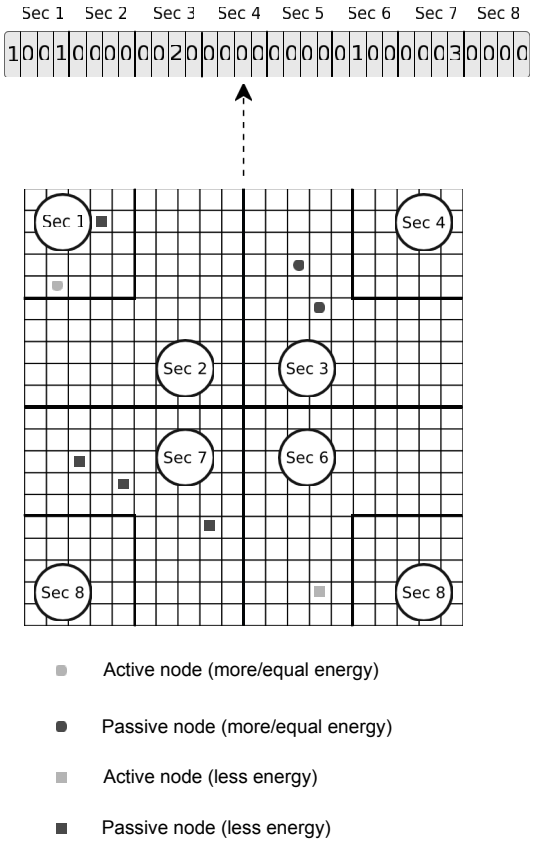


Figure 7.7: Encoding of the situation for ONC-controlled mode-selection protocols

(2) Similarity Metric Besides the pure representation of the current situation, the ONC system needs a possibility to compare situations and to determine their similarity. Considering the representation as depicted in Figure 7.7, the situation for ONC's learning component

is identical, if the sectors are e.g. shifted clockwise. More formally, this means that the possible influence of rotation and reflection have to be deducted before determining the distance by searching for the most similar match – similar to the MANet-based scenario before. Based on the most similar match, the absolute difference between all corresponding values of the situation description is added up and used as distance measurement. As a result, the distance value is lower for more similar situations. Consider as example that two situation descriptions are only differing in the representation of one sector: $Sector_{Sit(A)}$ is 1000 and $Sector_{Sit(B)}$ is 0000 – the distance measurement is 1.

(3) Learning Feedback As third step, the learning feedback is needed to enable automated learning. As mentioned initially, the main goal for mode-selection in WSNs is to reduce the energy consumption while simultaneously keeping the coverage of the area. Hence, the fitness function has to take three main aspects into account: 1) the distribution of neighbours within sending distance (goal: minimise number of active nodes), 2) the overlap rate of the environment (ensure complete coverage of the environment), and 3) the energy status (minimise energy consumption). All three aspects are measured locally at each node by taking the neighbours' status into account (available due to ADRA's mode-selection process). The heuristic formula is given as:

$$Fit(n) = \frac{x + y + z}{3} \tag{7.3}$$

with n identifying the currently observed network protocol instance, x representing the node availability part, y evaluating the degree of environmental coverage, and z determining the energy levels. This fitness value is calculated for each node locally, taking the covered area into account. The corresponding formulas for the three aspects are given as follows:

$$I.)\ x = e^{-(x_i-a)^2+b}; \quad II.)\ y = e^{-(y_i-c)^2}; \quad III.)\ z = d + e * z_i - f * z_i^2 \tag{7.4}$$

$a, b, \ldots, f$ are constants and have been configured empirically, whereas the resulting curves are chosen to achieve a specific behaviour. The x value as defined in part *I.)* of Formula 7.4 provides a Gaussian distribution for the interval between 0 and 100 % of active nodes within the node's sensing range. The goal of this aspect is to penelise situations where only a few nodes or nearly all nodes are active. Furthermore, the y value (see part *II.* of the previous formula) is defined between 0 and 100 % coverage rate of the environment. The resulting curve describes a normal distribution with a maximum at 100 % and drops strongly towards 0 when reducing the fraction of the actively covered environment. Here, the assumption is that uncovered parts of the environment lead to a decreasing detection rate. Finally, the z value defines the influence of the energy consumption of all nodes within sensing range on the fitness value. Based on the theoretical minimum and maximum values for the energy consumption and the number of nodes in sensing range, a ratio between

the determined energy consumption and the desired one can be determined. Therefore, a parabolic curve of the ratio is used, which penalises again the extreme values (all nodes being active or in standby mode) but favours low consumption values.

(4) Configuration Space As fourth task, ONC's control-interface has to be connected to the adaptable parameters of the ADRA-scheme. As discussed in Section 7.3.2, the authors of the protocol defined four basic parameter (*LocPrio*, *CovPrio*, *BatPrio*, and *Thsd*) that can be customised without interfering with the protocol's logic – these parameters will be subject to ONC control.

(5) Simulation Model Finally, a simulation model for Layer 2 of the architecture has to be provided. Since an optimisation of new parameter sets using a Layer 2-based simulation tool is not feasible for sensor nodes due to resource restrictions, Layers 1 and 2 are assigned to different locations (see Figure 7.8): Layer 1 remains on the sensor node, but Layer 2 is removed and transferred to a) the *sink node* and b) the *design time* for preconfiguration of the node's learning component. Since all nodes are identical, they can share the same rules: a rule that works for Node_A will show the same performance for Node_B in the same setting. For part *a)*, it can be assumed that long-lasting WSNs will have the possibility to distribute updates from the sink node to all sensor nodes using broadcast messages. Since rules for the learning component are simple pairs of situation/action and both can be serialised using only a few Byte and floating-point values with separators, the effort for communicating these new rules is low. Backwards, the communication of new needs as observed by single nodes can be performed by extending the already existing status and event messages by adding data about needed rules. In total, this means that an active management of the nodes' rule bases can be achieved without significant new overhead.

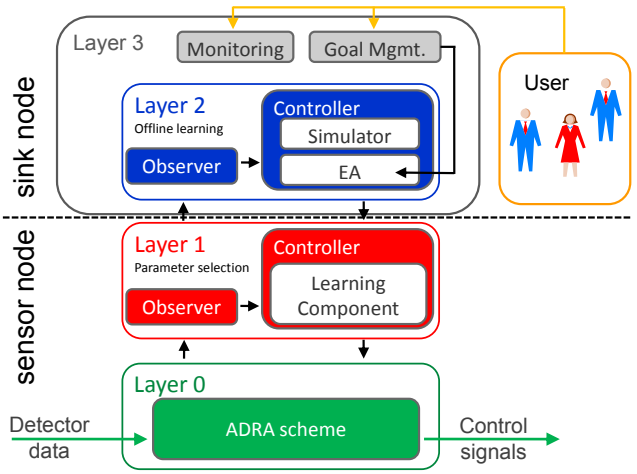


Figure 7.8: Adapted architecture of ONC for the application in WSNs

Alternatively and in addition to a sink-based update mechanism, the ONC system can be configured in advance to cope with arising situations at runtime (part *b*). Therefore, the simulation-coupled optimisation of rules is processed at design time. Hence, a trade-off between two extreme points is needed. On the one hand, a rule base can cover each possibly occurring situation using the sector-model. On the other hand, the computational effort and the needed storage increase linearily with the population size. Consequently, an appropriate size of the rule base has been determined incorporating as few rules as possible while still increasing the protocol's performance significantly.

This trade-off corresponds to the more general problem of finding a balanced solution between *adaptivity* aspects and effort due to *resource needs*. In current OC systems, it is usually assumed to have enough resources such that no trade-off is needed. But, especially in case of wireless sensor networks, hard restrictions for energy resources and computation power exist – the adaptivity is decreased for the benefit of making it easier to compute. In contrast, e.g. high-loaded routers in the Internet can be equipped with additional computation power in order to achieve a higher adaptivity and consequently increase the performance. Typically, the desired solution will be found between these two extreme positions – but again, this is application-specific and cannot be answered in general.

Experimental Setup

The experiments have been conducted using simulations. Therefore, the WSN environment has been implemented in JAVA using the *Recursive Porous Agent Simulation Toolkit* (Repast Simphony) [127]. The simulated area consists of 100 x 100 cells, which corresponds to 2,500 x 2,500 meters. In total, 150 nodes (149 sensor nodes and one sink node) have been created and equally distributed on this area. In order to imitate real-world installations, the distribution is chosen automatically at the begin of a simulation in a randomised way. The Physical/MAC layer is assumed to be an IEEE 802.11 in ad-hoc mode. Therefore, the sending distance is ten cells (corresponds to $250\,m$). As discussed before, the application to WSNs limits the available resources. Thus, the node's rule base has been restricted to 48 different classifiers, which corresponds to the best trade-off found in simulations. Especially compared to standard LCS-populations, this is an extremely small population size. The classifiers are the result of the off-line optimisation and selection process as mentioned previously – Layer 2 has been used to evolve matching parameter sets covering the space of possible situations. From this set, the best 48 classifiers have been selected. By increasing the population size, more specialised classifiers can be created leading to a potentially increasing system performance.

Like the authors of the ADRA protocol, an abstract representation of energy (the energy unit e) and its consumption has been used. Additionally, the Repast-based simulation relies on its own abstract unit of time, called *ticks*. A *tick* is equal to one step of the simulation. All nodes start with an initial energy level of $20,000\,e$. At each *tick*, all nodes consume

a defined amount of energy depending on their current mode: $5\,e$ in active and $0.5\,e$ in standby mode. Neglecting communication cost does not influence the results, as the effort is constant for all compared scenarios. The mode-selection algorithm is processed every $50\,ticks$.

Results of the Evaluation

In order to analyse the impact of ONC's dynamic adaptation process, a comparison between the static protocol version and the ONC-based version has been performed. In the following, three basic experiments are discussed: 1) a static distribution of nodes with periodically repeating events, 2) a static distribution of nodes with varying events, and 3) a dynamic distribution of nodes with varying events. The first experiment shows that the ONC system leads to a significant increase in terms of the performance metric, although the advantages of the approach compared to static versions are expected in more disturbed or dynamic settings. Thus, the second and third scenario are used to increase the degree of dynamics. In the second scenario, no pattern of repeating events can be learned anymore. Finally, the nodes have to deal with changing neighbourhoods of nodes in the third scenario – some will be disturbed during the simulation. The analysis is based on the same metrics as used by the authors of the protocol: the *network operation time* until the last node dies (no energy left) and the *observed events*. The goal of the system is to increase the network's operation time while simultaneously observing all occurred events. In addition, an aggregated fitness value is determined, which is based on the formula for the local learning feedback as introduced in Section 7.3.2. All figures depict the results obtained by 480 consecutive simulations using the same configurations – thus, the effect of the learning process is visualised. In this context, consecutive means that a run of the simulation is started with the rule-set of the previous run. Hence, the knowledge obtained by the learning component is transferred to the succeeding simulations.

In the first scenario, a static distribution of nodes and periodically repeating events (every 500 ticks) are simulated. Figure 7.9 illustrates the achieved results. The standard protocol version reported ten events (of 14 in total during the simulation time), corresponding to $7,241$ ticks until the last node of the network died. As depicted in Figure 7.9, the standard protocol version shows constant results for all 480 consecutive runs of the simulation. In contrast, the ONC-controlled version changes steadily due to the trial and update effects of the learning component – the network's operation time is increased by 4.4 % on average (corresponds to $7,556$ ticks). At each of the 480 consecutive simulations, the graph representing the ONC version is above the graph representing the reference version.

The learning effect is even more visible for the number of reported events. The ONC systems starts with a predefined set of classifiers, but it has to learn which one fits the current situation better. Until simulation run 285, the graph for the ONC-controlled version remains below the one for the uncontrolled version or oscillates around it. Afterwards, the number of

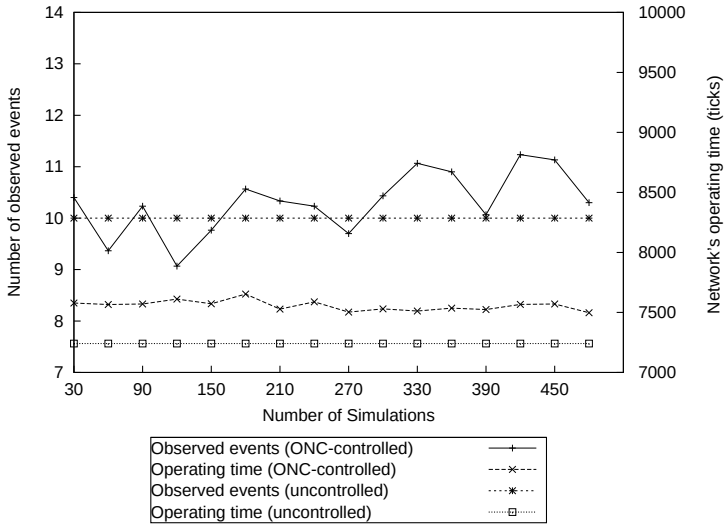


Figure 7.9: Comparison of uncontrolled and ONC-controlled system performance in a static scenario (higher values are better, zero is suppressed for both vertical axes)

successfully reported events is constantly higher. On average, the ONC-controlled version reported 10.3 events, which is an increase of 3 % – although, the initial learning period reduces the benefit for the complete simulation. Neglecting the influence of this intial learning period leads to a more significant increase. The network's *operating time* stays the same, but the number of *detected events* increases to 10.6 events.

The fitness function of the on-line learning component takes both previously discussed aspects into consideration. Figure 7.10 depicts the normalised fitness values. Since the fitness value of the standard configuration (uncontrolled ADRA scheme) stays the same for all 480 experiments, it is defined to be one in all cases – ONC's fitness value is then calculated in relation to this reference value. For the first static scenario, the fitness value of the ONC-controlled protocol instance has been increased by 3.7 % compared to the standard protocol configuration.

The results of the first, static scenario demonstrate the benefit of the additional ONC-control. The performance has been increased compared to the reference solution. In general, adaptivity as provided by ONC has its advantages in more dynamic environments. Therefore, the second scenario increases the level of dynamics. It keeps the setup of the previous one, but changes the event occurrence (randomised interval). Considering the results as depicted in Figure 7.11, the expected impact of the increased dynamics is clearly visible. Although there is only a small benefit of 4.6 % in terms of the network's operation time – the ONC-controlled version lasts $7,575$ ticks compared to $7,241$ ticks of the reference version,

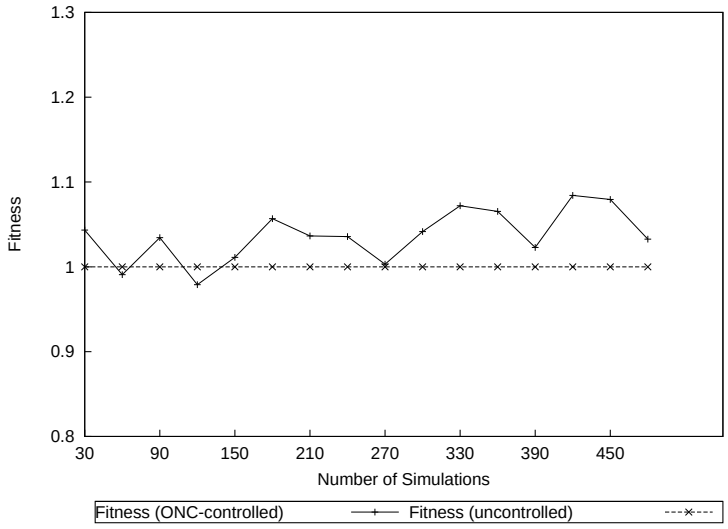


Figure 7.10: Achieved performance in the static ADRA scenario (higher values are better, zero suppressed for y-axis)

the number of reported events has been increased significantly. The ONC-controlled version reported 11.2 events on average, which is an increase of 22.2 % compared to the reference (nine events on average). Furthermore, the normalised fitness function shows similar results: 15 % (1.15 compared to 1 – see Figure 7.12) better than the reference.

The dynamics of the previous scenario have been modelled by randomly generated events. Considering real-world installations of WSNs, another dynamic factor is the distribution of wireless sensor nodes. Due to disturbances (e.g. caused by the weather), some nodes may fail during the network's operation time and other nodes may have short-term malfunctions. To simulate this behaviour, the third scenario provides an even more dynamic setting. In each run of the simulation, a set of nodes is randomly selected and disabled for a certain period. For this experiment, the following setup has been chosen: 18 nodes are selected randomly and disabled for 2,000 ticks. An undesired side-effect of disabling nodes is that they do not consume energy in this period and consequently increase the network's operation time. To counter this effect, disabled nodes consume the same amount of energy as defined for the standby mode. As depicted in Figure 7.13, the number of observed events can be increased by using ONC from 8 events to 9.3 events on average (16.3 % more). Simultaneously, the network's operating time is increased from 7,982 ticks to 8,321 ticks on average, which is a benefit of 4.2 %. The corresponding normalised fitness value results in an increase of 10.2 % (see Figure 7.14).

The achieved results for all three scenarios of the evaluation demonstrate the benefit

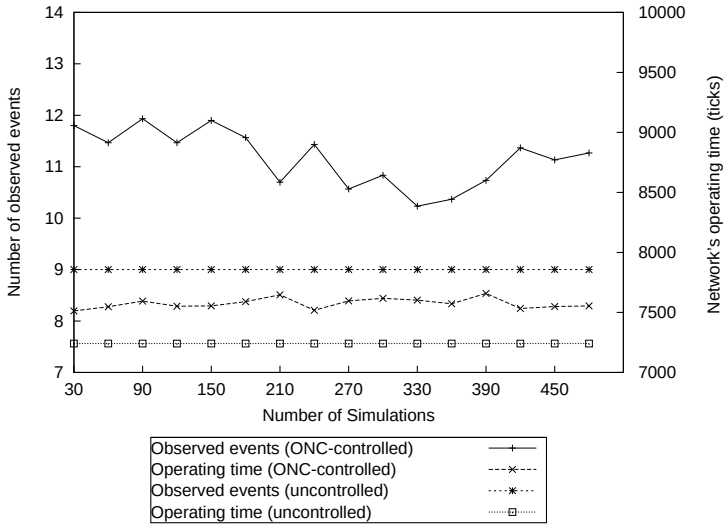


Figure 7.11: Comparison of uncontrolled and ONC-controlled system performance in the second ADRA scenario with dynamic events (higher values are better, zero suppressed for y-axis)

of using ONC for sensor nodes. The system's performance has been increased significantly in dynamic environments, but there is also a noticeable positive effect for static solutions. Especially when considering the hard restrictions applied to the ONC system, the results have been obtained with low additional effort. An LCS with just 48 classifiers is a very restricted variant of the learning part. Simulations with a rule-base of 250 trained classifiers applied to the first static scenario showed that the performance can be increased easily. Here, the network's operation time has been extended by about 25 % to 9,043 ticks on average. Thereby, the ONC-controlled version reported 14.1 events on average, which is an increase of 38.8 %. Considering the achieved results, the expected trade-off between adaptivity and resource usage has been observed.

Finally, the question arises whether the setup as described before can be applied to wireless sensor nodes. Considering the initially discussed representation of the situation, the condition part of a classifier contains eight sectors with four values encoded in Byte-format. In combination with four values encoding the action in floating-point-format, 48 $Bytes$ are needed for one classifier. The LCS at Layer 1 contains 48 classifiers, which means that a maximum of 2.5 $KBytes$ of RAM is needed. The system does not rely on having the whole rule base in RAM – instead, it can be stored in the node's flash-memory. The selection of a rule, based on the set of classifiers and the distance measurement as presented before, relies on 48 comparisons (number of classifiers) of 32 values in Byte-format (situation description)

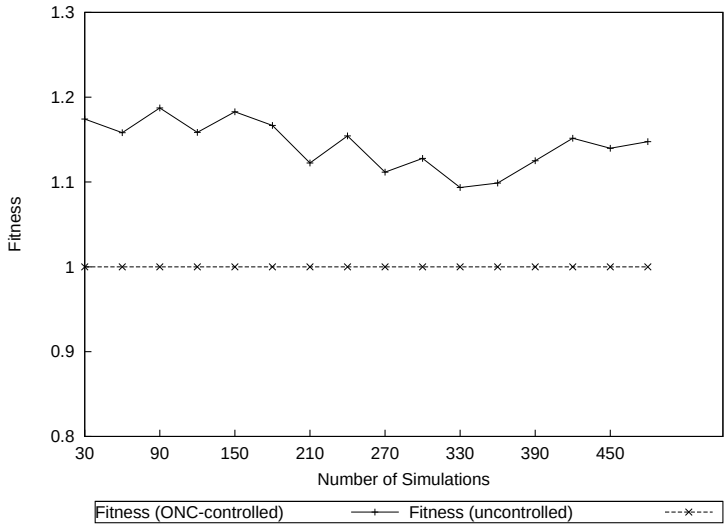


Figure 7.12: Comparison of the resulting fitness value for uncontrolled and ONC-controlled system performance in the second ADRA scenario with dynamic events (higher values are better, zero suppressed for y-axis)

– in total, about 1,536 simple Byte-comparisons have to be processed in each cycle. This is feasible for nearly all types of sensor nodes. Furthermore, an algorithmic solution of an adapted variant of *Wilson's* XCS has already been realised in hardware (see e.g. [19]) – which makes the application of ONC's Layer 0 and Layer 1 possible for wireless sensor nodes.

7.3.3 Peer-to-Peer Networks

The previous two examples have been used to prove the overall benefit of applying ONC to the control of a data communication protocol and to demonstrate the feasible application even under hard restrictions in terms of resource usage. Beyond these basic parts, the third scenario shows the limitations of ONC. Therefore, the Peer-to-Peer (P2P) protocol *BitTorrent* [174] is investigated which is widely-used and has been responsible for up to 53 % of all P2P traffic on the Internet [326] (based on measurements in June 2004). Recent studies estimate that P2P traffic comprises $40-70$ % of today's Internet traffic [327]. P2P protocols build an overlay layer on top of the physical network – therefore, the neighbourhood differs largely from the previous two scenarios. Since there are no physical neighbours having impact on the decisions of the particular node, all nodes currently known as *provider* of files (possessing the sought-after file) or as *consumer* (asking for files) form the neighbourhood. In

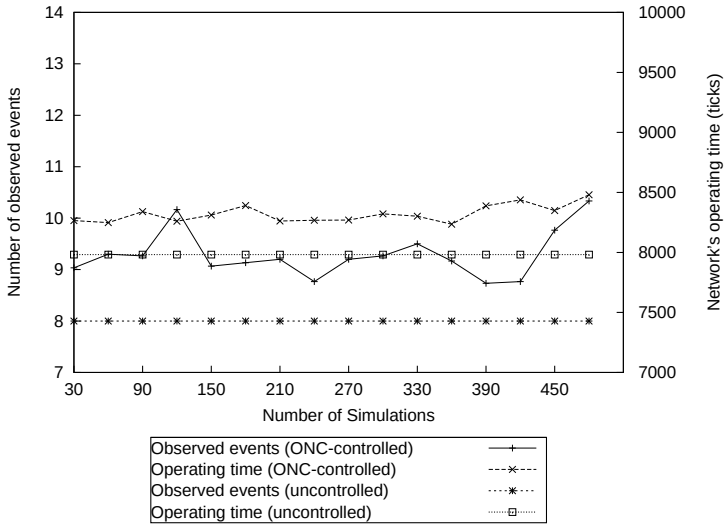


Figure 7.13: Comparison of uncontrolled and ONC-controlled system performance in the third ADRA scenario with dynamic events and random node-failures (higher values are better, zero suppressed for y-axis)

contrast to the scenarios before, a node has no knowledge about interdependencies between its neighbours – every node manages its own lists of peers. The local selection process for parameter settings can tackle this problem as it learns directly from the feedback and relies on existing rules. In contrast, the Layer 2 component needs a realistic simulation model, which is problematic due to the interdependencies between other nodes and the insufficient knowledge about the neighbours' status.

The investigation of this section is based on results already published in [282]. The section is organised as follows. Similar to the scenarios before, the presentation of the investigated protocol is followed by an introduction of the variable parameters controlled by ONC. Afterwards, the customisation of ONC is discussed in order to allow for a dynamic control of BitTorrent. Furthermore, the experimental setup is explained, which serves as basis for the achieved results presented afterwards. Finally, the limitations of applying ONC to P2P protocols are discussed.

Protocol

BitTorrent is a very popular and commonly used P2P protocol for distributing large files to a large group of users. Previous P2P protocols had problems regarding fairness or efficiency (like *Gnutella* and its relatives [328]). Thus, the developers focused on a fairness-based

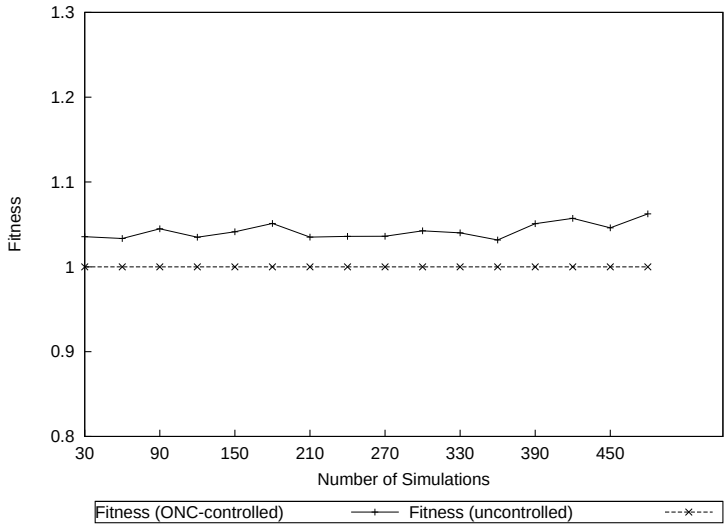


Figure 7.14: Comparison of the resulting fitness value for uncontrolled and ONC-controlled system performance in the third ADRA scenario with dynamic events and random node-failures (higher values are better, zero suppressed for y-axis)

distribution of data when developing the initial BitTorrent concept [174]. Therefore, the typically large files are split into small parts (so-called *chunks*). Besides the standard P2P client, the protocol contains an additional centralised web server (the *tracker*). This *tracker* is responsible for the bootstrapping and provides a search mechanism to find data files within the P2P overlay-network. A new client needs the address of such a tracker to become a member of the system, since this tracker manages lists of available parts for all files. When searching for a file, the peer receives a list of other peers providing chunks of the sought-after file. Furthermore, it can ask for more if the received set did not contain enough active peers. Individual chunks are exchanged following a tit-for-tat approach to achieve fairness and to avoid free-riding. A peer is only uploading data to those peers from which it receives the highest download rate; all other peers are said to be *choked*.

The process itself is organised by defining several parameters, which can be controlled by the peer (some only when setting up a new file for the overlay network, like the *chunk size*). Besides the choking mechanism, BitTorrent has another unique feature: the *optimistic unchoking technique*. Here, the client reserves a minor part of its available bandwidth for sending pieces to randomly selected peers with the intention of discovering even better partners and to ensure newcomers getting a chance to join the group. Further details on the protocol are given by *Cohen* [174].

Variable	*Standard value*
$NumberPeersPerTracker$	50 peers
$MinPeers$	20 peers
$MaxInitiate$	40 peers
$NumberOfUnchokes$	4 conn.
$ChokingInterval$	10 sec.
$RequestPipe$	5 req.
$MaxConnections$	1000 conn.

Table 7.3: Variable parameters of the BitTorrent protocol

Parameters

The process of the protocol is organised by defining several parameters, which can be controlled by the peer. One part of the parameter set refers to the tracker-based process when searching for files ($NumberPeersPerTracker$, $MinPeers$, and $MaxInitiate$), one part to the choking process ($NumberOfUnchokes$ and $ChokingInterval$), and one part to local request and connection settings of the client ($RequestPipe$ and $MaxConnections$). Besides these seven basic parameters, the protocol consists of more configuration possibilities. Since these further parameters are not accessible through the configuration interface of standard clients (like *Vuze* [329]) and ONC provides a black box solution, which does not interfere with the protocol's logic (other parameters are hidden within the protocol logic), they are neglected for the context of this thesis.

Table 7.3 lists the variable parameters of the BitTorrent protocol with their standard value as proposed by *Cohen*. The $NumberPeersPerTracker$ parameter defines the number of requested peers from the tracker, while $MinPeers$ specifies the minimum number of peers before asking the tracker for further peers. Additionally, $MaxInitiate$ defines how many peers are requested initially. Besides the request part, two parameters are used to configure the unchoking process. $NumberOfUnchokes$ defines the number of simultaneously unchoked connections and the $ChokingInterval$ specifies the time period between two verifications of the current choking status. Furthermore, the $RequestPipe$ defines how many requests of parts are sent to other peers simultaneously and $MaxConnections$ specifies the maximum allowable number of open connections to other peers (both, providers and requesters).

Customisation of ONC

Based on the same process as discussed for the previous two scenarios, the ONC framework is customised to the control of BitTorrent. Hence, the five basic tasks are discussed in the following part.

(1) Situation Description Again, the first task is to define a situation, which serves as input for the Layer 1 adaptation mechanism. Since the local BitTorrent client has no influence on other file-providing peers joining or leaving the network, an adaptation of the parameter settings according to the observed availability of other nodes is not feasible. Furthermore, the list of waiting peers to access files is managed locally by each peer – thus, nodes have no possibility to speed-up their waiting time. Considering these characteristics, the available influencing factors regarding download and upload of the BitTorrent client are the utilisations of local resources. For instance, visiting web-sites, streaming videos, or just using CPU- and RAM-resources for standard computational tasks have impact on the protocol's performance.

Typically, performing a network protocol is only affected by limited resources if the system is completely busy with other tasks. Within the following evaluation scenario, a normal course of day is simulated for a standard PC running an additional BitTorrent client. Thus, the resources to be monitored for the situation description are the available up- and download bandwidth. The corresponding goal for ONC is to utilise the free resources as well as possible by increasing the download rate. Hence, the situation description contains only the two values for the observed utilisation of up- and download capacity (without the fraction used by the BitTorrent client). Considering this situation description, insights on the limitations of ONC are visible already; all factors having impact on the node's current situation have to be fully accessible locally. Especially for protocols that are not based on physical connections, neighbourhood information can be transparent to the particular node.

(2) Similarity Metric For Layer 1, a measurement to compare the similarity of two situations is needed. The distance measurement can be formulated by calculating the Euclidian distance for a two-dimensional space. The two axes for the space defining the basis for the Euclidian calculation are given by the values for up- and down speed. For instance, the comparison of Situation_A (upstream: $25\,KByte$, downstream $100\,KByte$) and Situation_B (upstream: $20\,KByte$, downstream $80\,KByte$) leads to a value of 20.6.

(3) Learning Feedback As third aspect, a metric defining good and bad system behaviour is needed. As mentioned before, the goal of using a P2P client like BitTorrent is to download large files as fast as possible. Thus, the metric for the learning component is based on the achieved download speed of the BitTorrent client. Since the available bandwidth changes dynamically due to other tasks performed by the user, the relative accessible download speed is considered with $dlBit$ defining the download speed of the BitTorrent client, $dlCap$ the available download speed, and $dlUser$ the fraction required by the user:

$$Fit(x) = \frac{dlBit}{dlCap - dlUser} \tag{7.5}$$

(4) Configuration Space Additionally, the variable parameters to be controlled by ONC have to be defined. As discussed in Section 7.3.3, these are the seven basic parameters that are typically configurable using the client's control interface (see Table 7.3).

(5) Simulation Model Finally, a simulation model for Layer 2's sandbox learning component has to be provided. At this step of the process, the limitations of the ONC approach are visible again. A simulation model for P2P protocols has to take global information into account in order to provide a reliable simulated copy of the reality.

Every node in a P2P network keeps track of its own two lists: a) files to be downloaded and b) files to be shared. Furthermore, each link is individual due to the heterogeneity of the Internet and the varying usage profiles of the particular computers running the P2P client. But this information is not accessible locally – which makes a mapping from the real world to the simulator impossible. For the simulations performed in the next section, it is assumed that all nodes have the same characteristics. In particular, their network links (ADSL-line with 400 $\frac{Kbyte}{sec}$ for downstream and 40 $\frac{Kbyte}{sec}$ for upstream), resources (CPU and RAM), their resource usage (just the BitTorrent client), and the files (searched and provided) are equal. For instance, these assumptions are realistic when performing a laboratory setting with identical clients.

Based on these assumptions, a simulation model can be configured, which is applied to the discrete network event simulation tool NS-2 [130] again. *Eger et al.* developed a "BitTorrent-like" model of the protocol for NS-2 [175], which is available on-line at [330] and used for this scenario. The implementation is called "BitTorrent-like", because it does not implement a specific version of BitTorrent and relies on some simplifications. Although these simplifications lead to a decreased functionality of the whole protocol, it still behaves like the standard protocol – most of the abstractions are concerned with the distributed usage within the Internet. This simulation model is configured using the situation description obtained at Layer 0. The link characteristics are adapted with the values determined for the available up- and download bandwidth.

Experimental Setup

The evaluation shows the results of NS-2-based simulations. The considered scenario contains 100 peers and one tracker. Three peers are seeds: they have the complete file available from start on. All other peers do not have any part of the file. Each peer tries to download the same 500 MByte file during the simulation and starts over after finishing the download.

In this scenario, one of the downloading peers is equipped with an additional ONC system, all other peers are performing the standard BitTorrent protocol. The dynamic utilisation of the ONC-controlled node's network link is simulated by assuming an artificial usage-profile of a standard PC during one day. The user fulfils tasks, which leads to a certain utilisation of resources. Sporadic lookup of information and downloading of files blocks a

given fraction of the available up- and download capacity of the link, which is also used by the BitTorrent client running in the background to download a high amount of data. In the NS-2 simulation, this external usage of the link is simulated by reducing the available link capacity for the BitTorrent client. Thus, ONC's goal is to adapt the protocol configuration to the changes in the observed utilisation of system resources. The assumed usage profile is depicted in Figure 7.15.

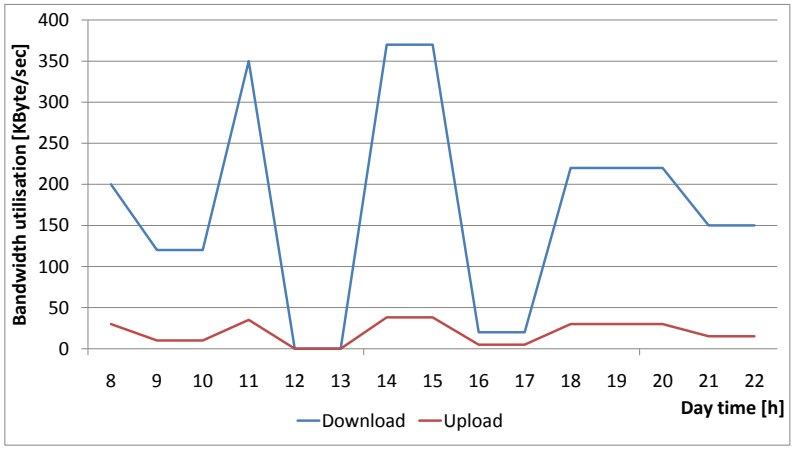


Figure 7.15: Assumed usage-profile of bandwidth during one day for the BitTorrent scenario

Results of the Evaluation

The evaluation compares the performance of the system (ONC-controlled and uncontrolled version) for three consecutive days. In this context, *consecutive* means that the simulation is started over after finishing by transferring the achieved knowledge. The existing classifiers from the previous simulation are available at the beginning of the next one. The results for all three days are depicted in Figure 7.16. During the first day, the system has nearly no knowledge available – it starts with an empty rule base and has to go back to the standard parameter set. Based on the observations, new rules are evolved and passed over to the rule base of Layer 1. The usage profile defines relatively static situations – only eight different situations occur during one day. Based on the similarity measurement as discussed for the adaptation of the ONC framework, only a few situations are similar enough to be used by the covering process. As a consequence, the system's performance at day 1 (average download rate of 165.5 $\frac{Kbyte}{sec}$) is equal to using the static standard protocol parameter configuration.

Considering the next simulated day, an improved situation-aware selection of parameter sets and the corresponding adaptation of the protocol can be observed. In comparison to day 1, the system's performance is increased by 7.2 % to an averaged download rate of 177.4 $\frac{Kbyte}{sec}$. Although the ONC system still does not have an appropriate rule for each observed situation, the desired effect can be observed. ONC recognises already observed

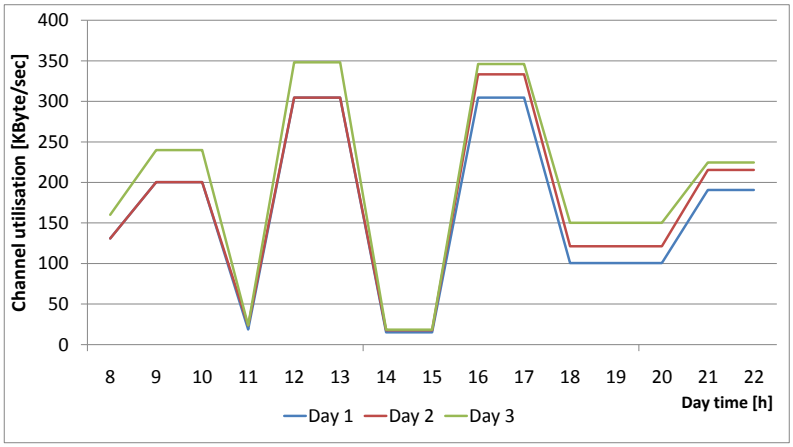


Figure 7.16: Resulting download performance due to ONC control of the BitTorrent protocol (higher values are better)

situations and uses the rules evolved by Layer 2. In some cases, a similar rule (*nearby* in terms of the distance measurement) is used by the covering mechanism (e.g. in the interval between 8 to 9 o'clock). Since Layer 1's controller is able to learn, it has to be allowed to choose not always the best-matching rule. With a lower probability, it tries a rule situated nearby (again in terms of the provided distance metric). This leads to the effect that the Layer 2-generated rule has not been selected in one case, although it was already available (see day 2, between 9 and 10 o'clock).

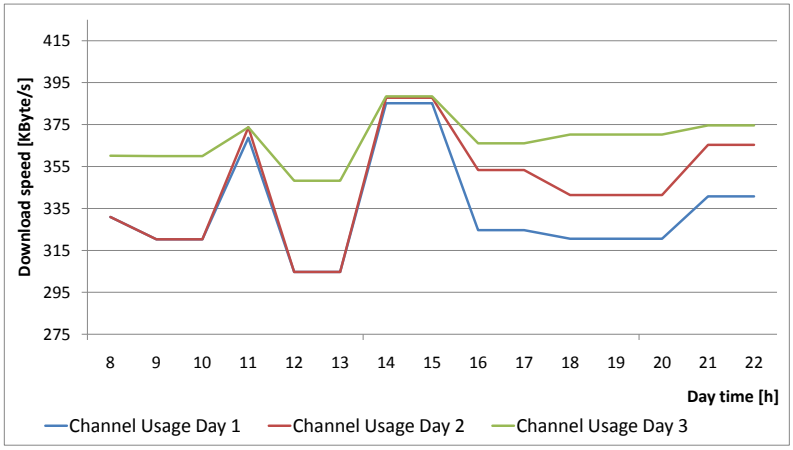


Figure 7.17: Aggregated channel utilisation for all three simulated days in the BitTorrent scenario (higher values are better)

The third day demonstrates the desired behaviour of ONC. It has already evolved an appropriate rule base that covers all occurring situations. The resulting averaged download rate is 199.3 $\frac{Kbyte}{sec}$, which is an increase of 20.4 % compared to the reference solution. This

demonstrates the potential of controlling networks by using ONC, since the static standard configuration does not lead to the best-possible performance. Finally, Figure 7.17 depicts the impact on the channel utilisation. The aggregated figures for both (BitTorrent under ONC-control and the assumed usage profile) are plotted to compare the development over three days.

In general, the scenario demonstrates the potential benefit when applying ONC to the control of BitTorrent and similar protocols. But it also reveals that the approach depends highly on the local availability of all attributes having impact on defining an appropriate simulation model and quantifying the system's performance. In the previous scenario, this is given only under certain restrictions. Apart from a laboratory-based setting, necessary information is missing. Alternatively, one could think about acquiring additional information by a protocol extension. Through this extension, peers would provide necessary information to the tracker, which in turn informs the particular peers. Besides legal and privacy issues, such an extension would cause a dramatically increased situation description – which would have influence on the classification of rules. A situation space covering the status of all known peers would not allow for a significant learning success in a manageable time frame. Thus, a control of BitTorrent by ONC seems to be only promising in cases where the network is restricted to uniform nodes – here, a significant increase of the performance is possible.

7.4 Collaboration in ONC

The performance of ONC depends strongly on the existence of *matching* rules. The learning component of the Layer 1 controller selects a matching (or a *nearby*, respectively) rule from the rule base and applies it to the SuOC. Currently, the rule base is extended using two mechanisms: the *covering* process and *Layer 2*. The covering mechanism provides a fast reaction by copying a nearby rule and adjusting the condition part to match the current situation – but a classifier created by covering is not an optimised solution. This optimised and situation-dependent classifier is generated by Layer 2 using simulations – the so-called *sandboxing*. Layer 2's most restricting limitation is the effort needed to evolve a new rule since simulations are time-consuming. For instance, the generation of a new rule in the first scenario (broadcast protocols in MANets, see Section 7.3.1) needs between one and up to 15 minutes depending on the scenario. Similar values can be observed for the WSN-based scenario of Section 7.3.2 (about 3 minutes on average) and the BitTorrent scenario of Section 7.3.3 (about 10 minutes on average).

Considering these time-demands, a start from scratch is problematic as it takes significant time and effort for a single node to evolve a sufficient rule base. A speed-up of the simulation leads to shorter evaluation cycles and a decreased simulation period for each parameter set under test – in turn, it affects the quality of the solutions. Thus, a promising approach

to achieve a speed-up is *collaboration*. Due to the similarity of nodes in terms of topology and characteristics, rules from one node can be re-used by another one. Furthermore, the Layer 2 component is identical for all nodes. Hence, one node can tackle an optimisation task for another one. Exactly these two aspects are the focus of the collaboration mechanism explained in the remainder of this section. Nodes will have the possibility to share knowledge with their neighbours by exchanging rules (and their experiences with these rules). In addition, the mechanism is used to realise a load balancing of Layer 2 tasks. The presented algorithm and the evaluation results have been published in [280] and are based on the implementation by *König* [331].

7.4.1 Dynamic Load Balancing and Knowledge Sharing for ONC

Due to the goal of reducing the number of rule generations by Layer 2 and of achieving a load balancing of rule generations among the nodes in the network, the approach intervenes at the interface between Layer 1 and Layer 2. According to the basic approach, Layer 1 is responsible for detecting the demand of new rules and reports this to Layer 2. Instead of immediately adding new optimisation tasks to Layer 2's queue, the distributed algorithm intervenes and takes over responsibility for an active management of this queue. The rule generation itself remains untouched – it processes always the queue's first entry. The remainder of this subsection explains how the distributed algorithm performs this queue management.

The distributed mechanism is a three-step process:

- In the first step, nodes observe the demand of a new rule and activate their collaboration unit to contact their direct neighbours.
- Once the necessary information has been transferred to the recipients, these neighbours check their local rule base and respond to the initial request.
- In the third step, the initial node analyses the answers and terminates the process.

At Layer 3 of the architecture, nodes are assumed to have the possibility of managing their neighbourhoods and of communicating with their direct neighbours. This assumption is extended: each node answers *immediately* to a received request. Hence, a short delay time after sending the initial request is enough to ensure an accurate feedback. Furthermore, non-collaborative or even malicious behaviour is neglected.

Part 1: Local activation Whenever Layer 1's LCS has no adequate rule for the currently observed situation, the collaboration mechanism is activated. So far, the basic version of the ONC framework (Layers 0 to 2) activates Layer 2 to evolve a new rule. Instead of directly activating its own Layer 2 component, the neighbours are involved into the rule-finding process. Therefore, a message-based technique has been developed guaranteeing a fast and correct exchange of knowledge. The message is composed of three parts: (a) the

situation description as basic setup information for Layer 2's simulator, (b) the origin node's current queue length, and (c) a TimeToLive-value (TTL) indicating whether the message has to be broadcasted to neighbours further away or not. The message is given as follows:

$$Message : \|HEADER|SituationDesciption|QueueSize|TTL\|$$

After sending the request to its neighbours, the node waits a given delay time (while the neighbours are processing part 2) and continues with part three. The delay interval can be determined as follows: $2 \times (Maximum\ sending\ duration) \times TTL + (processing\ constant)$. If no neighbours are in transmission range, the task is directly added to the queue of Layer 2.

Part 2: Process request The second part of the process is performed by all neighbours receiving the message from part 1 of the process (directly or forwarded from another node). As the TTL-part is handled by lower protocol layers, the main task of the nodes is to check whether they know an adequate rule for the submitted situation or not (current tasks within the queue are also taken into account). If a node has a matching rule, this is sent to the inquirer. Otherwise, the node compares the received queue size with its own. For those cases where the received queue size is higher than the own one, the next spot in the queue is reserved for the task and the inquirer is informed. The corresponding spot in the queue is blocked for a time period t_{block}. If the own queue has more tasks than the inquirer's queue, a reject is returned.

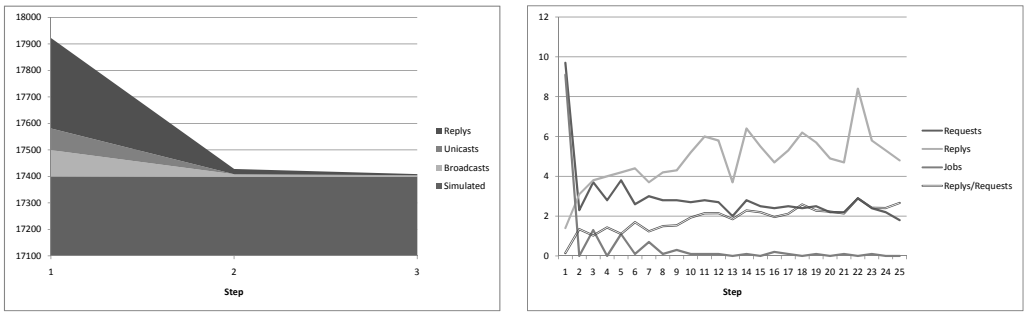
Part 3: Finalise mechanism Finally, the node that started the process is able to complete the last part. Therefore, it analyses the received answers. The approach distinguishes between two different possibilities: a) the node received a rule and b) it received no rule. In the first case (a), this rule is added to the rule set of Layer 1 and the process terminates. Otherwise (b), the received offers have to be analysed. The process accepts the best offer and informs the respective node. If no acceptable offer is received, the task is added to the own Layer 2 queue.

7.4.2 Evaluation of the Collaboration Mechanism

The collaboration mechanism has been evaluated using the MANet-based broadcast scenario from Section 7.3.1. The setup of the evaluation is based on the *NS-2* scenarios developed by *Kunz* in [276] for the initial protocol. The first scenario contains 100 nodes and analyses the load balancing performance. All nodes are using their ONC system, but only ten nodes are able to evolve new rules by using their Layer 2 (they are assumed to have a better power supply). If a node has no *active* node in its direct neighbourhood, it can increase the TTL-value to consider e.g. a two-hop neighbourhood. The initial rule base for each node is empty, which means the ten active nodes have to evolve new rules for all 100 nodes. All results are averages of ten runs. The scenario is processed in three consecutive steps:

- All 100 nodes determine their current situation (empty rule base).
- The collaboration mechanism takes place.
- The process is validated, i.e. the experiment is repeated with the newly acquired knowledge provided by the learning mechanism.

The learning mechanism was able to respond correctly to a previously learned situation resulting in the same performance at both occasions. Since the population of the classifier system is initially empty, the standard parameter sets are used for the performance evaluation in step 1. In step 2, the queues of all ten *active* nodes have been processed and the rules have been distributed. The fitness calculation takes place after processing each step. The overall fitness (same function as in Chapter 7.3.1) of the system is 0.882 in the first step and increases to 0.926 in the second step (equal to step 3). On average, each active node needs 3 hours and 42 minutes to complete the queue of optimisation tasks (the maximum is 7 hours and 6 minutes). Due to the abstraction of situations (some situation descriptions are similar enough to be handled in the same way), an averaged number of 94.3 jobs has been performed in total. Each active node had to process 9.43 jobs on average (the most heavily loaded node had to process 11.33 *jobs*). The different queue sizes can be explained by the distribution of nodes – two active nodes had only a few neighbours (lowest: five), while the other eight had to work for comparatively more nodes (up to 14).



(a) Messages in the first evaluation scenario

(b) Messages in the second scenario

Figure 7.18: Evaluation of the Collaboration Mechanism in ONC

The second part of the experiment determines the effort in terms of exchanged messages. Figure 7.18(a) illustrates the results. The amount of system-wide messages changes during the three consecutive steps: the total number of messages drops from 17,922.6 to 17,400 on average in step 3. An averaged number of 522.6 messages has been simulated for the collaboration mechanism (Broadcasts 99.6, Unicasts 81.9, and replies 341.1) – which is just a small overhead compared to the 17,400 BCast messages (2.9 %).

Finally, the nodes' reactions on abruptly changing situations have been investigated. Therefore, a static setup of the scenario without node movements has been chosen and nodes were exchanged. Nodes are not moving to avoid the occurrence of new situations and

force them to receive the particular rule from their neighbours. The exchange takes place by randomly choosing two nodes and switching their positions. The scenario consists of 25 steps and contains again 100 nodes, the broadcast distance is two hops. Initially, all nodes are active nodes (they learn the matching action for their particular situation). Since nodes are not moving, the environment is static and consequently the observed situations stay the same. Within each of the 25 steps, ten pairs of nodes are randomly selected and their positions are exchanged. Probably, these nodes have to receive a matching rule due to the changed situation – although their Layer 2 is deactivated (at least the predecessor at their position has an appropriate rule). Figure 7.18(b) demonstrates the message traffic for all steps. The graphs for the number of jobs decreases nearly to zero, which means that almost every exchange is followed by a transmission of the rule. In some cases, a job is performed, which means that the exchanged pair is not available in a two-hop neighbourhood. The number of responses (and also the reply/request ratio) increases continuously as nodes do not delete rules. Hence, the goal of the collaboration mechanism is fulfilled, since the exchange of rules works and a re-creation of rules is avoided.

7.5 Summary for the Organic Network Control System

This chapter discussed the **Organic Network Control** (ONC) system, which has been developed to allow for a context-aware adaptation of network protocol parameters. The system is based on the thesis' framework and has been applied to three different protocol domains ranging from *Peer-to-Peer* protocols over *mobile-ad-hoc networks* to *wireless sensor networks*. Based on a discussion of related work, the application of ONC to these three scenarios has been explained with a special focus on the customisation tasks identified in Chapter 3.

In contrast to the OTC example, a unified process to enable ONC control for communication protocols is not possible due to e.g. the varying neighbourhood information. Therefore, the application has been shown exemplarily by means of three different protocol types. The first example demonstrated the positive effect of additional ONC control for a MANet-based broadcast algorithm. The analysis of the obtained results highlighted the possible benefit and investigated the necessary effort to achieve these results. In general, the scenario described a case where ONC can be applied without any restrictions and leads to a significantly better performance.

The second example has been used to investigate the applicability of ONC to protocols under limited availability of resources. Therefore, a mode-selection protocol from the domain of wireless sensor networks has been chosen, since wireless sensor nodes are typically characterised by the demand of energy-efficient solutions. Especially computational-intensive tasks like simulations cannot be performed at such nodes. Thus, a separation of tasks has been proposed: Layer 1 stays at the particular node while Layer 2 is transferred to a cen-

tralised element. The approach relies on a pretraining of rules at design time and an optional update mechanism using communication at runtime. In addition, the LCS's rule base has been largely restricted to using only a few classifiers. Although this setup describes a trade-off between adaptation and available resources, the simulation-based evaluation showed the benefit of using ONC on sensor nodes. The two most important figures (the *network's operation time* and the *event detection rate*) have been improved significantly – especially in disturbed and dynamic situations.

The third scenario applied ONC to the control of the Peer-to-Peer protocol *BitTorrent* and has been chosen to investigate the limitations of ONC's applicability. In comparison to the previous two examples, BitTorrent is characterised by limited knowledge about the neighbourhood and the current local status of the network. The simulation-based generation of new rules depends on the availability of a realistic simulation model, which has to be configured appropriately using the observed attributes. In case of BitTorrent, a peer has only limited knowledge, which does not allow for building an appropriate simulation model except in laboratory settings where peers are identical. Besides these restrictions, the evaluation demonstrated that ONC control of BitTorrent leads to a significant increase of the system's performance in such a laboratory setting.

Finally, the simulation-based generation of new rules has the drawback of needing computational effort and being time-consuming. Thus, a collaboration mechanism has been developed which is capable of sharing existing knowledge and enabling load balancing of Layer 2 tasks among neighbouring nodes. The evaluation demonstrated that the fully decentralised approach leads to the desired effects. Therefore, the first scenario of MANet-based broadcast algorithms has been used. Compared to the OTC example, ONC has the advantage that all nodes are identical – thus, they can re-use rules from their neighbours and will probably gain the same experiences. In contrast, intersections of urban road networks differ strongly in their particular topology (apart from artificial Manhattan-type networks) – consequently, rules cannot be transferred to neighbouring nodes in OTC. When applying the mechanism to OTC, only the load balancing part could be used since simulation models might be part of the messages.

Chapter 8

Generalisation and Discussion

The previous two chapters presented systems based on the framework. Both provide a substantial contribution to current research in their particular domain. Besides developing novel concepts for data communication and urban traffic control, this thesis focuses on the general applicability of the developed framework. Thus, the following chapter aims at generalising the approach and discussing the achieved results. Therefore, the applicability of the framework is proven by considering different control problems. In contrast to the network-based systems before, this chapter introduces two further application scenarios: an *Organic Production System* (OPS) and an *error-prediction system for business mainframe systems*. In the first example, the control problem differs from OTC and ONC as it discusses a task-assignment problem. In the second example, the focus of the sandbox has been exchanged: from a simulation-coupled optimisation heuristic to a data-mining-based prediction component.

The previously named examples make the general applicability of the developed concept visible. In order to prove this applicability and to derive insights for which control problems the framework leads to the desired behaviour and for which more probably not, the SuOC is abstracted by using mathematical models. These abstractions are useful to describe different types of control problems and their corresponding degree of complexity. In addition, the impact of noise added to sensor data can be measured, since the undisturbed values are known for comparison. Finally, the achieved results of all application scenarios including the more general mathematical model are summarised and discussed.

8.1 An Organic Production System

Industrial production experienced a dramatic change in the last 20 years, which is characterised by novel manufacturing technologies and by the rapidly growing importance of information and communication technologies. Nowadays, production differs largely from previous approaches. On the one hand, much effort has been spent on optimising setup and schedule of production systems. On the other hand, flexible and adaptive solutions become even more important. Both aspects are mainly driven by economical needs to keep production cost at a minimum. Due to this continuous optimisation process, researchers and industry have developed a large set of different solutions ranging from highly specific installations to more generalised concepts with a broader focus.

The aspect of needing even more adaptive solutions makes production control a promising application domain for OC principles and techniques. In some settings, new tasks arrive continuously and sometimes even unpredictably, while a production without corresponding demands leads to full warehouses and high cost. Therefore, a continuous observation and control process is needed responsible for deciding about the best possible situation-depending production strategy automatically. Based upon the architecture described in this thesis, a case study for an **Organic Production System** (OPS) has been developed. The results have been published in [332]. The framework has been adapted by exchanging the underlying control problem. Instead of continuous parameter values, the system has to decide about the currently best task assignment for the available production units. In order to restrict the complete domain of production control to one exemplarily investigated scenario, and to map the previously described problem to a specific application case, the following setup has been developed. Consider a fast-food store where the food is produced on-demand as response to customer demands. The store has a set of production places where workers prepare the different products. Each product has a certain production time, and changing of tasks leads to a given changeover delay. In addition to the production part, a storage for prepared products exists where the products can be kept for a certain duration until they are sold or disposed. Hence, products are assumed to have a relatively short *shelf-life* and cannot be stored for a longer period. Simultaneously to the production process, a counter exists where customers can drop their orders and wait to be served.

The evaluation of the scenario shows that the framework is applicable to such a scenario. In case of taking advantage of a production forecast (enabling the prediction module of the Layer 1 observer), the OPS system outperforms the reference solution in terms of customer satisfaction (measured by considering the customers' waiting times) and defected goods (sorted out products at the end of the shelf-life). Without prediction, the system increases the performance for one part of the goal function only. For further details on the scenario and insights on the results, the reader is referred to [332].

8.2 An Error Prediction System for Mainframes

Based upon the architecture described in this thesis, *Li* developed an automated system to predict undesired events in an *IBM System Z* [333] setting [334]. She adapted the framework by exchanging the Layer 2 component. The simulation-based approach has been replaced by a data mining mechanism, which analyses observed real-world data from customer servers. The goal has been to find patterns like a specific sequence of events before an error occurs. Thus, the task of Layer 1 is not to adapt a SuOC, but to serve as an early warning system at runtime. Therefore, a *Frequent Episode Mining* technique as introduced by *Mannila et al.* [335] has been applied to event series. The patterns found by the data mining technique are stored as classifiers for the Layer 1-situated LCS, which is realised as the modified variant introduced in Chapter 4.1.4.

The different types of customer data were collected on-line at the customer side and then transmitted to a central server. *Li* tested the developed system using such real data from customer servers – both *error-free* and *error-prone*. She demonstrated the reliability of the system's prediction for different settings and analysed the influence of the on-line learning part on the one hand and the performance of the off-line pattern recognition on the other hand. Since the work presents a case study and a proof-of-concept of the described approach, further investigations are needed. Nonetheless, the approach underlines the generic character of the architecture.

8.3 Abstraction of the Learning Problem

In the previous examples, concrete problems have been solved by applying the developed framework to a specific application domain. Most of the investigated examples provide a significant contribution to the state of the art in the particular domain. So far, the application of the framework has been compared to realistic models of real-world installations since the sizes of the underlying situation and configuration spaces do not allow for making a statement in relation to the possible optimum. In the following, this question is investigated using an artificial scenario based on introducing an abstracted model of the underlying problem. Due to this abstraction, it is possible to draw conclusions about the applicability of the developed framework. The contents of the following section have been initially published in [336] and the results are based on the implementation as done by *Brameshuber* [337].

8.3.1 Problem Description

The basic problem covered by the following part of the thesis is given by the question for which types of real-world systems the developed approach is applicable – and in which situation problems might occur. The basic tasks of Layers 1 and 2 are to find adequate response

strategies for observed situations and to learn a mapping between situations and appropriate actions. Thereby, the term *situation* describes an n-dimensional space defined by all observed input variables defining the state of the environment, while actions correspond to parameter settings of the SuOC. Hence, the learning component has to achieve two goals: (1) find the best action for an observed situation and (2) increase the accuracy of predicting a certain feedback if this action is selected (typically called *reward*). More formally, it has to learn the following mapping: $(\vec{x}_t, \vec{z}_i) \rightarrow r(\vec{x}_{t+1}) \in \mathbb{R}$ with the objective to choose $\vec{z}_i$ to maximise r. Thereby, $\vec{x}_t$ denotes the n-dimensional situation vector at some time t, which corresponds to the condition part of Layer 1's action selection. $\vec{z}_i \in Z$ aggregates a parameter set with Z describing the SuOC's configuration space. Furthermore, the predicted *reward* of the next evaluation time $t+1$ (if $\vec{z}_i$ is chosen in situation $\vec{x}_t$) is represented as a real value and denoted as $r(\vec{x}_{t+1})$.

8.3.2 Application of the Generic Architecture

In order to investigate the success of the developed framework, a generalised model of the learning problem is introduced. The goal of this generalisation is to consider several types of models to map situations to corresponding actions. Simultaneously, it must be possible to calculate the optimal action of the learning component for each occurring situation, which allows for quantifying the success in relation to the best possible solution. Figure 8.1 describes the adaptations at design level, which are introduced in the following part of this section.

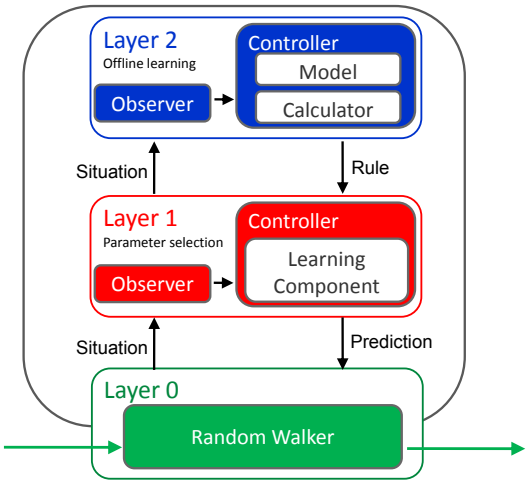


Figure 8.1: Modified architecture for the generalised model

Modelling the Environment: The environment is modelled as an n-dimensional space, which corresponds to the n observed attribute values obtained by sensors to de-

scribe the current situation. The situation at some time t is one point in this space and defined as the vector $\overrightarrow{x}_t$. In dynamic environments, the situation changes constantly, i.e. any of the n entries in the vector may vary from time t to time $t+1$. These dynamics are modelled by a randomised walker exploring the situation space. In order to reflect the assumption that situations in real-world applications are only changing abruptly in the presence of disturbances, the n-dimensional situation $(x_1,\ x_2,\ \ldots,\ x_n)$ at time t is modified by a random walker to create the following situation $(y_1,\ y_2,\ \ldots,\ y_n)$ at time $t+1$ with $-1 \leq (y_i - x_i) \leq 1$:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

The size of the allowed deviations in each dimension corresponds to the speed at which the random walker traverses the search space. In the investigated scenario, this speed is set to one for each dimension. Three different movement models for the random walker have been implemented: 1) a classic random walker, 2) a random-waypoint model, and 3) a random-direction model (see e.g. [321]). In the first model, the walker follows the current direction within the situation space determined by the difference of the last two situation vectors: $(\overrightarrow{x}_t - \overrightarrow{x}_{t-1})$. The walker decides about modifying its current movement direction by considering a Gaussian-distributed probability for the interval $[-180°, 180°]$. As soon as it reaches the boundaries of the search space, the movement direction is reflected by the boundary of the search space. In the second model, the walker chooses a random point within the boundaries of the investigated search space (the *waypoint*) as next destination. Afterwards, it selects the next available point, which reduces the distance to this waypoint in each step of the simulation, until it arrives at the waypoint – then, the previous process is repeated. Finally, the last model chooses a direction of movement, which is followed until the end of the search space is reached. In this case, a new direction is chosen randomly and followed until the previous criterion is fulfilled again.

Layer 0 – System under Observation and Control: In the basic design, the SuOC is the productive part of the system and adapted to the particular environmental situation. This means that an optimal parameter setting exists for each occurring situation. This relation is modelled by using a mathematical function mapping the situation space on the corresponding optimal SuOC configuration. The resulting correct actions are determined by different functions with varying characteristics. According to *Cakar et al.* [338], systems can be classified into three categories considering the time-dependent character of the learning problem's fitness landscape: *static*, *dynamic*, and *self-referential*. The *static* class corresponds to all systems where an action $\overrightarrow{z_j}$ in a situation $\overrightarrow{x_i}$ results always in a repeatable and constant reward. In contrast, this reward changes for systems in the *dynamic* category, e.g. according to time-dependent patterns. Even more dynamic are the rewards of systems

in the *self-referential* class – here, the system adaptation to match the observed situation influences the next situation and thereby the received reward. In the context of this thesis, the regard is confined to systems of the *static* class (see Chapter 3.2). Within this class, four different types of functions modelling the optimal action for a given situation space can be distinguished:

- **Type 1** describes a continuously differentiable function without any local maxima mapping the situation space on the action space – this trivial type is seldom found in reality. The following function serves as example:
  $f_1(\overrightarrow{x}) = \sum_{i=1}^{n} x_i$.

- **Type 2** comprises more complex continuously differentiable functions being characterised by more than one maximum. The following function serves as example:
  $f_2(\overrightarrow{x}) = (1 + sin(\frac{x_1}{4} * \pi)) * \sum_{i=1}^{n} x_i$.

- **Type 3** classifies even more complex non-continuously differentiable functions with discontinuities (not due to time-dependent reasons), etc. – at least, these mappings can be modelled using functions. The following function serves as example:
  $$f_3(\overrightarrow{x}) = \begin{cases} \sum_{i=1}^{n} x_i, & \forall x_i : (x_i \neq 5) \\ \sum_{i=1}^{n} \sqrt{|x_i|}, & \text{otherwise} \end{cases}.$$

- **Type 4** covers all randomised mappings – if no relation between the action values of two neighbouring situation values exists, a functional description is not possible. To investigate such an example, a random function value for each possible situation value has been generated and stored in a database:
  $f_4(\overrightarrow{x}) = Random_{Seed}(\overrightarrow{x})$.

Layer 1 – On-line adaptation: In contrast to the Layers 0 and 2, Layer 1 is identical to the standard implementation – in particular, it does not have any knowledge about the form of f. The modification affects only the purpose of the on-line adaptation, since no parametrisable SuOC exists in the abstracted model. Therefore, the functions of type 1 to 4 are used to describe the best "configuration" in a particular situation. Hence, the environment and the SuOC are collapsed into the function f. Figure 8.2 illustrates this modification of Layer 0. Analogously, Layer 1 has to determine this *best configuration*. Thus, the actions performed by Layer 1 are used to predict the next value of the underlying function (as defined by type 1 to 4). Thereby, not the functional value for $\overrightarrow{x_t}$ observed at the current time step t is to be predicted, but the value that is going to occur at time $t+1$. This corresponds to the same learning problem as in e.g. OTC where the best possible phase durations for the next evaluation period are selected based on (averaged) traffic data from the previous one. The Layer 1 controller is implemented as modified LCS again (see Chapter 4.1.4).

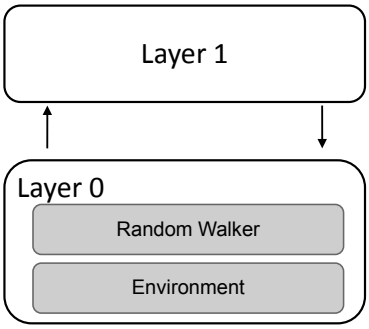


Figure 8.2: Modifications at Layer 0

Since the covering part of the modified XCS relies on quantifying similarities between situations (*nearby*), a distance metric is needed. In this example, the Euclidian distance of two situation vectors has been used. Additionally, a learning feedback is needed to quantify good and bad predictions - the *reward*. In the OTC and ONC examples, this reward is calculated based on sensory input or internal attributes. To model such an observation, Layer 1 is assumed to be able to observe the correct functional value in the following time step. Based on this value, the modified LCS can update its *prediction*, *prediction error*, and *fitness* values according to the original XCS algorithm by *Wilson* [159].

The improvement of the prediction of the next functional value is influenced by two components: a) the rule-generation process and b) the updating of the classifiers' performance attributes. Therefore, a *reward* has to be defined quantifying good and bad performance. The goal of the system is to decrease the deviation of the predicted functional value over time. Therefore, the reward $r(t)$ obtained from the SuOC at time t is modelled as follows:

$$r(t) = 1 - \frac{|PredictedValue - OccurredValue|}{max(|PredictedValue|, |OccurredValue|)} \tag{8.1}$$

In case of a perfect prediction, the reward is one, while it decreases to zero in the worst case. This feedback is calculated by the SuOC and provided together with the situation vector in each step of the simulation.

Layer 2 – Rule generation: The framework's basic design contains a simulator-coupled optimisation heuristic for Layer 2, since usually no exact model exists to determine the optimal response for a given situation. This optimisation takes time, but it is assumed that this process results in near-to-optimal actions for the particular situation. In OTC and ONC, new rules are usually available after 3 to 15 evaluation cycles of Layer 1 depending on the configuration of Layer 2 (simulation time, number of evaluation function calls). To imitate this delay, a value at the lower boundary of this range is assumed (3 ticks). Simulations showed that choosing longer delays has only impact on the shape of the achieved performance curve (the system converges later), but does not influence the learning effect in general. Similar to the realistic examples, Layer 2 is only allowed to optimise one rule

per time. Since an accurate model exists and the evaluation aims at analysing the learning behaviour of Layer 1 rather than Layer 2, the function used for the SuOC is assumed to be available at Layer 2. Thus, no simulation-based optimisation is needed. It is consequently replaced by a calculation of the particular function for the observed situation vector.

The following list summarises the five customisation tasks for the application of the framework to the abstracted learning problem:

(1) Situation Description: The current position of the random walker in the search space represents the situation description. This corresponds to the n-dimensional vector $\overrightarrow{x_t}$.

(2) Similarity Metric: The Euclidian distance is used as distance metric between two situations.

(3) Learning Feedback: The learning feedback reflects the success of predicting the correct functional value for the next time step $t + 1$. Therefore, the formula as defined by Equation 8.1 has been developed.

(4) Configuration Space: In contrast to the previous application scenarios, the abstract model does not contain a parametrisable system. Hence, no configuration in the direct sense is searched. Therefore, the prediction of the functional value of f occurring in the next evaluation cycle $t + 1$ serves as replacement of this configuration task.

(5) Simulation Model: Since the correct model exist (i.e. f is known), no simulation-based optimisation is needed. This process is modelled by determining the correct functional value for the particular situation description and providing it with a certain delay.

8.3.3 Evaluation

The evaluation of the developed approach has been performed using a 2.66 GHz dual-core machine with 4 *GByte* RAM and openSUSE 10.3 as operating system. Each result is given as average of ten runs using varying random seeds. For all investigated scenarios, the number of dimensions has been set to six and the boundaries of the search space for each dimension to $[0, 50]$. The modified LCS has been configured according to *Butz and Wilson* [162].

Experiment 1: Impact of the Different Walker Models

The first part of the evaluation considers how changing situations influence the learning performance by taking the three different walker models into account. Figure 8.3 shows the results using function type 1 and the three different walkers. The figure depicts the simulation time (x-axis) and the achieved learning effect (the deviation between the predicted value by the modified LCS and the correct function value; y-axis). The simulation time has been measured in terms of *ticks*, which refers to an abstract time unit used e.g. in Multi-Agent Toolkits such as *MASON* [128] or *Repast* [127]. One *tick* corresponds to one simulation step. Considering the figure, the different walker models have only limited

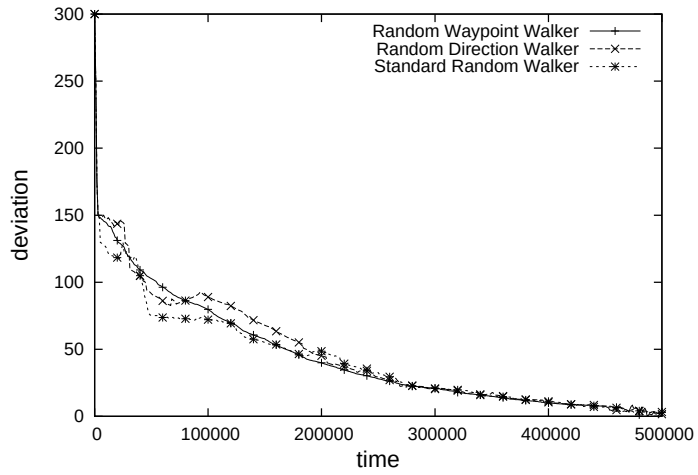


Figure 8.3: Comparison of the system performance caused by all three random walker models (lower values are better)

influence on the learning performance. For the complete simulation period, the *Random Waypoint Walker* resulted in an averaged deviation of the predicted functional value of 43.78, the *Random Direction Walker* of 46.23, and the standard *Random Walker* of 41.94. Besides the similar convergences of the learning effects using all three walker models, the values after $500,000$ ticks simulation time are nearly identical – the averaged values of the last $5,000$ ticks are: 2.22 (Random Waypoint), 2.37 (Random Direction), and 2.83 (Random Walker). The similarity of the results obtained by the three different walker models has been observed for the other function types (f_2, f_3, and f_4), too. Thus, the initial assumption has been supported: it does not matter how the situation changes as long as it changes with a minimal similarity to the previous situation. Therefore, the remaining part of the evaluation will be based on using only one walker model: the *Random Waypoint model*.

Experiment 2: Different Functions

The previous experiment compared the three walker models using only one of the four function types. In contrast, Figure 8.4 depicts the results achieved by using one walker model for all four types of functions. In total, function type 1 results in an averaged deviation of the prediction of 43.78, f_2 results in a deviation of 69.64, f_3 in a deviation of 86.87, and f_4 in a deviation of 159.96. Statistically, a randomised guessing without any learning effect would lead to predicted values of 150 on average – this value is deduced from the number of dimensions multiplied by the centre of the search interval:

$$randomDeviation : 6 * 50/2 = 150 \tag{8.2}$$

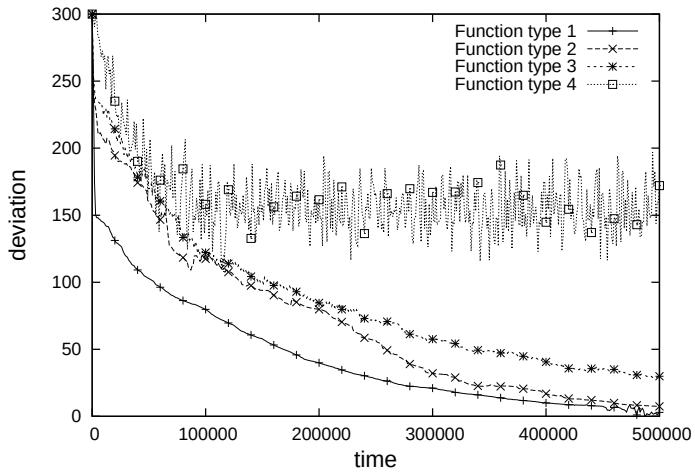


Figure 8.4: Comparison of the system performance taking all four different types of functions into account (lower values are better)

Correspondingly, the maximum deviation is given by $6 * 50 = 300$: the number of dimensions multiplied by the highest possible deviation per dimension. Obviously, the performance of the learning system based on a situation-action mapping modelled by function type 4 is comparable to randomised guessing. In contrast, the first three types of functions lead to a convergence due to learning: f_1 converges to values of about 2.22 on average after $500,000$ ticks simulation time. f_2 (7.50) and f_3 (29.11) show a similar behaviour, but result in higher values. In general, the system is able to self-improve the selection process of the corresponding actions in case of a possible modelling using types 1 and 2 without any restrictions. In case of function type 3, the self-improvement is restricted, since it leads to a significantly higher deviation.

Experiment 3: Impact of Noise

The four types of functions f_1 to f_4 define basic categories – each type can be further classified according to the observation characteristics: either it is *noisy* or not. Typically, real-world applications rely on the usage of sensors to observe the environmental and the internal states. Sensors are error-prone and to some degree unreliable. Thus, these applications usually have to cope with noisy feedbacks. Therefore, this part of the evaluation considers *artificial noise* by taking a Gaussian-distributed random noise-factor into account when determining the current functional value for the n-dimensional situation vector – this noise is assumed to amount to at most 10 % of the absolute value. Thus, the third part of the evaluation investigates the impact of noisy feedback according to this simplified concept. Figure 8.5 illustrates the results. The first graph describes the behaviour of the proposed framework again using the modified LCS and a Random Waypoint Walker dicovering the

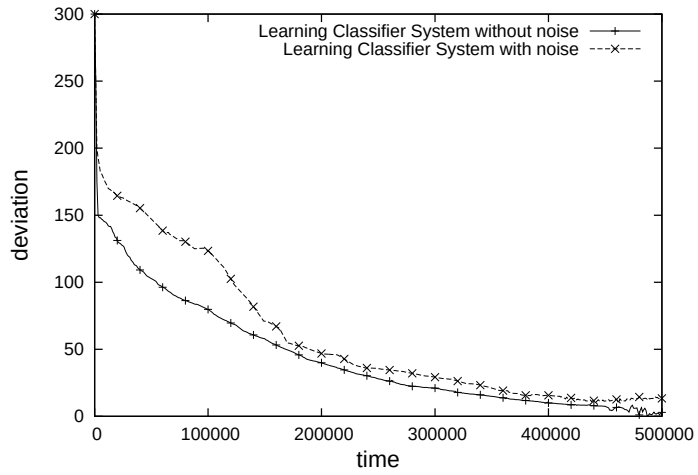


Figure 8.5: Impact of noise on the system's performance (lower values are better)

situation space. The corresponding actions are modelled using the function of type 1. The second graph describes the results obtained for the same setup using the additional noise model. Considering both curves, a later convergence and a slightly higher deviation after $500,000$ ticks of simulation time are visible. In general, the noisy setup resulted in an averaged increase of 36.6 % for the complete simulation period (59.82 instead of 43.78). The averaged values of the last $5,000$ ticks are: 2.22 (without noise) and 13.79 (with noise). Thus, noise has influence on the absolute performance, but it has only a minor effect on the convergence itself.

Experiment 4: Alternative Rule-selector

For comparison reasons, a deterministic rule-selection mechanism has been developed for Layer 1 which does not incorporate on-line learning capabilities. This *SimpleSelector* replaces the modified LCS and relies on using the same distance metric for the selection process. It chooses always the *nearest* rule (by means of comparing the situation parts of the rules). Similar to the modified LCS, it queries Layer 2 to receive new rules in case of unknown situations. Thus, this *SimpleSelector* is used to distinguish between the *on-line* and the *off-line* learning effect, since it relies only on knowledge from the latter mechanism.

Figure 8.6 depicts the obtained results for the modified LCS variant and the *SimpleSelector* approach. In both cases, the Random Waypoint Walker and the action modelling using function type 1 have been used again. Considering the figure, the *SimpleSelector* leads to better results at the beginning. This can be explained by the trial-and-error parts of the LCS system – the usage of non-optimal rules in the particular situations resulted in a later decrease in terms of deviation of the predicted correct functional value. Considering the complete simulation time, the LCS-based approach resulted in an averaged deviation

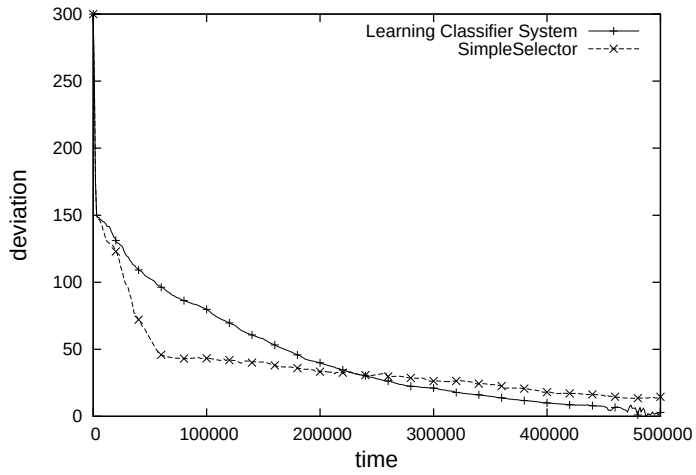


Figure 8.6: Comparison of the achieved performance for the standard LCS-based system and a SimpleSelector-based alternative (lower values are better)

of 43.78, while the SimpleSelector resulted in an averaged deviation of 37.46. In total, the SimpleSelector showed a better performance, but this effect decreases with longer simulation times.

Starting at tick 230,000, the learning effect becomes visible. The performance of the LCS-based approach is better than the SimpleSelector-based solution. The SimpleSelector chooses always the nearest rule, since it is not able to generalise situations based on the received feedback. As a result, it makes less mistakes at the beginning compared to the LCS. The SimpleSelector does not converge to zero, but to a constant value of about 13.5. This is caused by the configuration: for both techniques, a minimum similarity has been defined. The Layer 2 component is only triggered if the rule base does not contain a rule whose situation part is at least ten units away from the observed situation (using the Euclidian distance). Without this limitation, the rule base would become too large, since the system would query a new rule for each occurring situation. Hence, the SimpleSelector should converge to this configuration value. In contrast, the LCS has the possibility to generalise existing other rules – this part of the learning process causes the better values at the end of the simulation period.

Experiment 5: Impact of Disturbances

The last part of the evaluation introduces *disturbances* to imitate severe challenges of real-world systems. In reality, such disturbances can be malfunctions of system components (like sensors) or just unanticipated behaviour. Mapping this concept on the model to explore the situation space, a disturbance is defined as an unanticipated change of the walker's position. Therefore, two kinds of re-positioning effects are distinguished: a) only one dimension of the

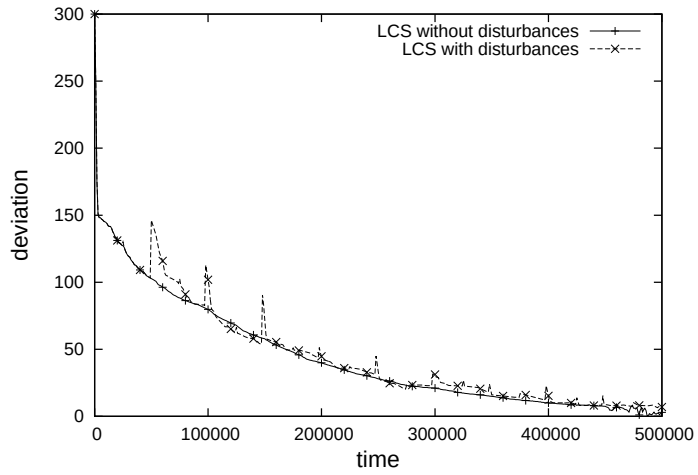


Figure 8.7: Impact of disturbances on the system's performance (lower values are better)

situation vector is affected (e.g. one sensor is disturbed) and b) all dimensions are affected (unanticipated behaviour). In the former case, a randomly selected dimension value is replaced by a new random value (within the given boundaries of the search space); in the latter case, this is done for each dimension. Both types of disturbances occur several times during the simulation leaving enough time in-between to allow the system to return to normal system behaviour. To avoid regularity effects, an alternating occurrence has been chosen which takes place in a Gaussian-distributed interval of ± 50 ticks every $25,000$ ticks of the simulation time.

Figure 8.7 depicts the obtained results for the *disturbed* and *undisturbed* cases. The scenario covers again the LCS-based learning approach, the Random Waypoint Walker and the function type 1. Considering the graph with the disturbed values, the disturbances are visible due to the peaks every $25,000$ ticks. In one case (simulation time between $108,000$ and $150,000$ ticks), the disturbed curve shows a better performance than the undisturbed one. Analysis of the obtained simulation data showed that in these cases the previous disturbances resulted in earlier knowledge, which has been used within this period. In general, such effects are expected to disappear when taking more simulations into account for calculating the averages. The performances caused by both approaches are comparable – the disturbances lead to isolated effects, which can be covered quickly. In addition, the averaged influence of the disturbances on the deviation decreases, since the available knowledge covers similar situations and therefore reduces the undesired disturbance effects relatively faster. In total, the disturbed variant resulted in a slightly increased deviation over the complete simulation period (46.8 instead of 43.7 – about 7.01 %). Thus, the assumed behaviour that the developed system covers disturbances without significant effects on the overall performance has been demonstrated.

8.3.4 Application of the Developed Classification

The previous part of this chapter introduced a novel classification for OC systems and applied the developed framework to representatives of each class. Thereby, it has been shown that the framework is able to control systems belonging to type 1 and 2 of the classification, while type 3 can only be controlled under certain restrictions. Consequently, the question arises to which of these classes the previously described applications have to be assigned to – namely the OTC system (see Chapter 6) and the ONC system (see Chapter 7). Hence, this question is exemplarily investigated for the former application in the remainder of this section.

In general, some statements about the characteristics of fitness landscapes (and thereby about the mapping function f) can be made for real-world systems. As one example, *Reeves* observed that "[...] on average, local optima are very much closer to the global optimum than are randomly chosen points, and closer to each other than random points would be. That is, the distribution of local optima is not *isotropic*; rather, they tend to be clustered in a *central massif* [...]" [195]. This observation is also referred to as the *big valley theorem*. In the context of this thesis, the most important point in *Reeves'* work is that fitness landscapes of real-world systems are mainly characterised by the same attributes and shapes. Furthermore, *Kauffman* investigated the characteristics of real-world fitness landscapes in general and introduced the *N K-landscapes* [196] – they also describe the same effects as observed by *Reeves*.

Hence, we can assume that the underlying fitness landscape tends to model similar functional dependencies for the investigated application scenarios. This is a first indication but does not lead to concrete results for the considered OTC scenario. Thus, the corresponding function to map the situation to the configuration space is needed as a basic requirement to investigate the question to which type of the classification OTC belongs. In contrast to the other examples (like ONC or OPS), such a mapping can be modelled for OTC by using one of the existing heuristic approaches to approximate traffic delays for traffic situations. Based on the occurring delays, the performance of the traffic control strategy can be quantified (cf. Chapter 6.3).

Typically, the formulas by *Webster* [245] and *Akçelik* [339] are used in traffic engineering to estimate delays – in the context of this thesis, the former formula is focused. According to *Webster*, turning delays can be derived as follows. Assuming that M corresponds to a turning's current traffic flow (in $\frac{veh}{h}$) and that SF denotes the turning's saturation flow (i.e. the maximal flow that can be served assuming permanent green), the turning delay t_d can be computed as defined in Equation 8.3. Thereby, SF is a constant and typically chosen as $1,900$ for standard urban intersections. Furthermore, t_C denotes the intersection's cycle time, while t_g represents the turning's effective green time (both given in seconds). In addition, $g = M/(t_g/t_C \cdot SF)$ defines the degree of saturation of the turning for the current traffic flow and green time.

$$t_d = 0.9 \cdot \left[\frac{t_C \cdot (1 - t_g/t_C)^2}{2 \cdot (1 - M/SF)} + \frac{1800 \cdot g^2}{M \cdot (1 - g)} \right]. \tag{8.3}$$

Equation 8.3 calculates the delays for one turning. To estimate the average vehicular delay for a signalised intersection, the delays for all turnings of the intersection are combined in a flow-weighted sum. Assuming that M_i is the traffic flow for the i-th turning and that $t_{d,i}$ specifies the turning's delay computed according to *Webster's* formula, the average delay t_D of the intersection can be calculated as defined by Equation 8.4. This equation serves as evaluation function and metric for the quality of a mapping from situation to action.

$$t_D = \frac{\sum_i (M_i \cdot t_{d,i})}{\sum_i M_i}. \tag{8.4}$$

The exemplary investigation of the OTC scenario is performed by assuming a four-armed intersection similar to the one depicted in Figure 6.3 – although the particular topology does not have any noticeable effect on the following example (which means that the approach works as well for different kinds of intersections). Furthermore, the scenario assumes constant traffic conditions and green durations for all turning movements except turning 1 (see *Turning A* in Figure 6.3 – but which specific turning is chosen does not have any impact). In addition, the following Table 8.1 list the assumed traffic conditions for each turning and the assignment to the corresponding phases. Afterwards, Table 8.2 describes the assumed green time durations for the particular phases. In addition, $P1$ to $P3$ denote the phases 1 to 3, while $I1$ to $I3$ denote the corresponding interphases in Table 8.2.

Turning	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
Flow ($\frac{veh}{h}$)	300.0	50.0	200.0	50.0	300.0	50.0	50.0	300.0	200.0	50.0	300.0	50.0
Phase	P2	P2	P1	P3	P3	P3	P2	P2	P1	P3	P3	P3

Table 8.1: Assumed traffic example for the intersection of Figure 6.3

Phase	P1	I1	P2	I2	P3	I3
Duration	28 *s*	2 *s*	28 *s*	2 *s*	28 *s*	2 *s*

Table 8.2: Assumed phase duration for the example of Table 8.1

Based on this setup, the functional relation between different green-phase durations for **one** turning, the different amount of traffic for this turning, and the corresponding evaluation criterion as defined in Equation 8.4 can be estimated. Figure 8.8 illustrates the achieved results.

Considering Figure 8.8, an assignment of OTC to type 2 of the classification is possible. The figure demonstrates that the relation between an occurring situation and the possible actions (as response from the control mechanism define by Layers 1 and 2 of the architecture) show the characteristics as required by type 2: the mapping is continuously differentiable and leads to a fitness landscape with *one* global optimum and optional local optima. The

figure illustrates the relation between situation, action, and evaluation criterion for varying the situation and configuration space of **one** dimension of the corresponding vectors. This limitation to one dimension is caused by the restrictions of visualising the results in a three-dimensional figure. Taking more dimensions into account does not affect the observations – it leads to more local minima and maxima, but it still results in a continuously differentiable function without any discontinuities. Hence, OTC belongs to type 2 of the classification.

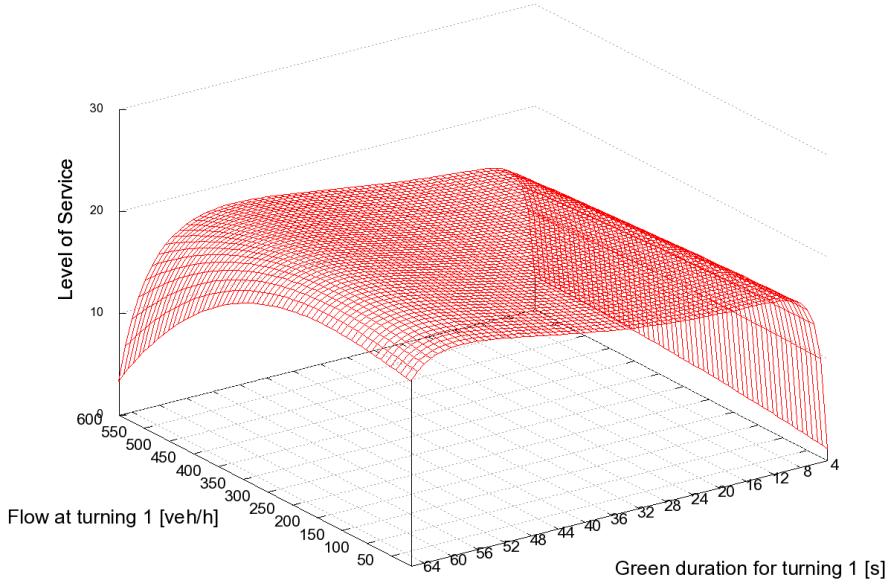


Figure 8.8: Three-dimensional plot of the mapping between situations, actions, and corresponding evaluation values in the OTC example

For data communication protocols, no approximation functions to derive a measurement for the quality of the solution exist. Hence, it is not possible to validate the type of classification analogously to the aforementioned case. But there are some indicators and similarities suggesting that ONC also belongs to type 2 of the classification. Similar to OTC, the parameters are continuously configurable. In addition, previous work [278] investigated the relation between situations, actions, and the corresponding fitness evaluation of the MANet-scenario (cf. Chapter 7.3.1). The result of this investigation is that the fitness landscape models a continuously differentiable function with local maxima/minima. Consequently, ONC is also assumed to belong to type 2.

In contrast to the OTC and ONC examples, OPS is characterised by a different control problem and hence by a slightly different functional mapping. Instead of adapting continuous parameter values, the adaptation mechanism has to decide about discrete task

assignments. Due to these "either – or" decisions, each change of the configuration vector causes discontinuities in the fitness landscape. This is a strong indicator that OPS belongs to type 3 of the classification.

8.4 Discussion

Based on the initial design of the framework, several application scenarios have been presented in the previous chapters. These applications range from traffic control (see Chapter 6) over three different implementations of network protocol control (see Chapter 7) and a case study for production control (see Chapter 8.1) to an error prediction system in mainframe systems (see Chapter 8.2). In addition, an abstracted model to analyse the general applicability of the developed concepts has been presented. This abstracted model is based on a proposed classification introducing four different classes of real-world applications. Applications can be assigned to a certain type by considering the relation between the situation space and the corresponding best configuration as one point within the configuration space (see Chapter 8.3).

Especially based on the last example, statements about the general behaviour of the system are possible. Hence, the analysis of the general model showed that applications of type 1 and 2 are controllable by the framework. This statement holds for type 3 applications only with major restrictions. The approach itself is applicable and leads to a stabilising system behaviour, which is characterised by a significant improvement compared to arbitrary or static configurations. But for the investigated cases, a qualitatively clearly worse performance has been observed compared to the types 1 and 2. In contrast, applications of type 4 cannot be controlled by the framework. But this was to be expected, since a random function contains no information in terms of a predictable regularity. Furthermore, the reaction of the system has been investigated when exposed to noisy rewards. The general behaviour of the system defined by the developed framework can cope with a certain degree of noise (in the investigated scenario the noise made up for up to 10 % of the reward). In addition, it is robust against these disturbances that do not cause the system's situation and configuration to leave the defined boundaries of the corresponding spaces.

Besides the general applicability of the framework to different types of real-world systems, the possibilities and limitations of the proposed system have been focused when investigating the particular application scenarios. Therefore, the wireless sensor network scenario of the ONC system served as example for highly limited resources (see Chapter 7.3.2). Typically, nodes in sensor networks are characterised by major restrictions considering the possible effort for computation and communication. Here, the simulative results demonstrated the positive effect even for small rule bases and the absence of Layer 2. In contrast, the BitTorrent application of the ONC system served as example where the real-world restrictions resulted in a less powerful application of the framework (see Chapter 7.3.3). Due to missing

knowledge about the network's status, an appropriate simulation model cannot be derived from current observations. Consequently, an integration of ONC into existing BitTorrent clients had no positive effect for the protocol's performance. In contrast, the way how situations change seems to have no influence on the system performance (cf. the different walker models in Chapter 8.3.2) – as long as a certain similarity between consecutive situations can be guaranteed.

In addition, some exemplary modifications of the sandbox component have been presented. On the one hand, the component has been used only at design time due to the previously mentioned resource restrictions in wireless sensor network settings. On the other hand, the simulation tool has been replaced by an approximation function (the *Webster* formula, see Chapter 6), or the complete component has been exchanged by a data mining concept (see Chapter 8.2). Although the particular realisation of Layer 2 differs, the purpose is still untouched – it has to generate new rules based on safety restrictions and without affecting the live system. Considering these different possibilities, several further approaches might be promising depending on the particular scenario. As far as available for the underlying scenario, analytic solutions might be applicable. Furthermore, an empiric consideration of the basic search space might be possible and consequently makes the optimisation component redundant. Especially for real-world systems, the application of both approaches is not realistic due to the typically vast situation and configuration spaces. The opposite direction is to extend the system's degree of freedom – in particular for those cases, where only inappropriate simulation models exist or necessary knowledge is missing (i.e. the BitTorrent scenario of ONC). In these cases, Layer 2 might be exchanged by a component with a modified focus. Instead of finding the best-possible action for a given situation, it might verify or guide the selection process of Layer 1. For instance, this component can restrict the set of selectable rules for the on-line system according to previous experiences. Simultaneously, Layer 1 receives a higher degree of freedom by e.g. using the non-modified XCS as proposed by *Wilson*. For further thoughts in this direction, the reader is referred to [340].

Another important question arises when applying automated learning to technical systems: does the mechanism converge to stable solutions within a reasonable time frame? This issue has been investigated in cooperation with *Rudolph* and the results have been presented in [341]. This work considers the convergence of the modified variant as used in the framework as well as the original XCS algorithm. It is proved mathematically that such a convergence can be guaranteed under certain restrictions (i.e. choosing a constantly decreasing learning rate).

Summarising the previous discussion, the applicability of the developed framework and the potential benefit of the resulting adaptive and self-organised solution have been demonstrated. Based on the different application scenarios and the generalised model, the scope of the framework has been defined and analysed. Thus, the initial goal of the framework has been fulfilled.

Chapter 9

Conclusion

This thesis presented a framework to enable Organic Computing principles like adaptivity and robustness for existing technical systems. Based on the general design of the framework, design questions about techniques to be used for particular tasks have been investigated and answered. Afterwards, the framework has been applied and customised to different domains. In particular, the control of traffic lights at urban intersections and the dynamic reconfiguration of data communication protocols have been investigated and analysed in detail. Furthermore, two case studies considering adaptive production control and the prediction of errors in mainframe systems have been presented followed by introducing an abstracted model of the underlying learning and control problem. Based on this model, the question for which kind of systems the framework is applicable has been answered by considering a classification of real-world systems.

The previously described contents of this thesis contain a set of contributions to research's state of the art in the following domains:

- **Organic Computing / System design:** A system architecture to allow for considering adaptivity aspects at design level has been developed. This architecture describes a generalised way of equipping existing parametrisable systems with desired characteristics as proposed by Organic Computing. In addition, a framework to realise the general design has been presented, which serves as customisable black box solution for engineers wanting their application to be characterised by additional attributes like adaptivity, robustness, and self-improving behaviour.
- **Machine learning:** Two modified variants of existing machine learning techniques have been developed and presented in the context of this thesis. Both techniques – a modified Learning Classifier System and a modified Fuzzy Classifier System – allow for on-line learning while simultaneously taking real-world restrictions into account.

In addition, the behaviour of the Learning Classifier System-based approach has been analysed using a generalised model of the underlying learning problem of real-world applications. Accordingly, these real-world systems have been classified using the introduced model.

- **Traffic Control:** The Organic Traffic Control system describes a novel approach to traffic-responsive control of traffic lights at urban intersections. Due to its decentralised installation, the demand-oriented collaboration, and the self-improvement at runtime, it can cope with problems arising in currently used concepts.
- **Adaptive network protocols:** Situation-aware reconfiguration of network protocols has gained increasing attention in the last decade. The Organic Network Control System proposes a novel approach to achieve situation-aware networking by proposing a fully decentralised solution based on the developed framework. The approach has been applied to three exemplary protocols: a reliable broadcast algorithm for mobile ad-hoc networks, a mode-selection protocol for wireless sensor networks, and a Peer-to-Peer protocol.

Although describing a generalised approach to adapt existing systems dynamically according to changes in the environmental conditions, the presented work leaves space for further research. For the general framework, this concerns questions about the following issues:

- **Further Organic Computing characteristics:** In the present thesis, the major focus has been set on enabling adaptivity and robustness for systems to be controlled by the developed framework. Besides these fundamental Organic Computing characteristics, several further aspects exist. For instance, the aspect of **flexibility** has been outlined at a technical level when introducing the modified Learning Classifier System, but it has not been analysed in detail. This topic is currently investigated by *Becker* in his diploma thesis [165]. In addition, the model calibration is a second issue, which has been mostly neglected in the context of this thesis. Currently, the simulation model of Layer 2 is configured according to the observed situation. But what happens, when the reality changes such that the model itself becomes inappropriate? Further investigations could focus on developing concepts to determine these changes and consequently adapt the simulation model.
- **The purpose of Layer 2:** As discussed in Chapter 8.4, the design of the sandbox component can differ for varying application scenarios. Especially in cases where the simulation-based solution has its limitations, alternatives are needed. One promising approach is to increase Layer 1's degree of freedom and to use Layer 2 for guiding the decision process of Layer 1 (see also [340]).
- **Optimisation heuristic:** Recent work in the context of Organic Computing introduced a novel search heuristic, which is especially designed to deal with noisy envi-

ronments: the *Role-based Imitation Algorithm* [338, 342]. Thus, it suits perfectly to the demands for the Layer 2 component. Current work shows that the Role-based Imitation Algorithm outperforms the chosen Evolutionary Algorithm in one Organic Network Control-based example.

- **Layer 3:** In the current scenarios, Layer 3 has been investigated exemplarily by developing ad-hoc solutions. Examples are mechanisms to establish Progressive Signal Systems in urban traffic networks or to achieve a load balancing of Layer 2 tasks. A detailed investigation of the layer and a generalised model of the component have not been the subject of this thesis.
- **Active learning / exploratory behaviour:** Typically, Layer 2 is heavily loaded at startup of the system due to mostly empty rule bases. Afterwards, the work load decreases dramatically. The resulting idle times can be used to actively explore the underlying search space in order to generate knowledge for situations that might occur later on. Alternatively, the system can try to find better solutions for situations where the system's performance has shown to be imperfect.
- **Undesired feedback effects:** In the current version, each node equipped with the additional framework learns autonomously and adapts its parameter configurations according to the locally most-promising strategy. The evaluations have shown that such an approach has positive effects on both, the single node and the complete network. But uncoordinated local decision can lead to feedback effects and oscillating behaviour caused by local decisions and learning – in Organic Computing typically referred to as *emergent effects*. Currently, these effects are not considered.
- **Social components:** The system design takes a decentralised coordination and collaboration at the highest layer of the architecture into account. Especially current trends in Organic Computing propose to extend the scope by taking normative and social elements into consideration (see e.g. [11]). Thus, it might be useful to extend Layer 3's focus.
- **Trust:** The locally organised collaboration between neighbouring systems equipped with the developed framework is currently assumed to work in a perfect world. In particular, all nodes are assumed to work together, to serve the overall goal, and to fully cooperate. Thus, misbehaviour or undesired demeanour are neglected. When moving the systems from simulations to the real world, safety concepts have to be considered, which makes principles like *safety* and *trust* necessary.

Besides these general considerations, an application to further technical domains can be promising. In addition, moving individual applications like Organic Traffic Control and Organic Network Control from simulations to the real-world could lead to novel insights on the concept's applicability and help to formulate ideas for upcoming demands.

Organic Traffic Control For the Organic Traffic Control system, the overall goal should be to push the development towards real-world applicability in urban traffic control. Thus, the main topics for further work are closely connected to problems arising when moving the application from simulation to the real world: connect the developed system with actual sensor and control equipment, provide interfaces for current installations, and set up test cases using traffic lights in protected environments. Probably, the less development-based aspects of clarifying legal and liability questions are even more important.

Organic Network Control The Organic Network Control system has been analysed using single, isolated protocols only. Typically, data communication needs a set of different protocols for varying purposes when transmitting data from a sender to a receiver. Thus, cross-layer or multi protocol optimisation can be seen as natural next step in the development process. In addition, the investigated protocols are mainly characterised by a limited number of variable parameters and a restricted application area. In general, most of today's traffic is processed using the UDP/TCP and IP protocols of the Internet – successfully applying Organic Network Control to this general protocol stack would finally prove the applicability and possible benefit of Organic Network Control.

Bibliography

[1] Jeffrey O. Kephart and David M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[2] Holger Prothmann, Fabian Rochner, Sven Tomforde, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck, "Organic control of traffic lights," in *Proceedings of the 5th International Conference on Autonomic and Trusted Computing (ATC-08)*, Chunming Rong et al., Eds. 2008, vol. 5060 of *LNCS*, pp. 219–233, Springer Verlag.

[3] Roy Sterritt, "Autonomic Computing," *Innovations in Systems and Software Engineering*, vol. 1, no. 1, pp. 79 – 88, 2005.

[4] Sven Tomforde, Emre Cakar, and Jörg Hähner, "Dynamic Control of Network Protocols - A new vision for future self-organised networks," in *Proceedings of the 6th International Conference on Informatics in Control, Automation, and Robotics (ICINCO'09)*, Joaquim Filipe, Juan Andrade Cetto, and Jean-Louis Ferrier, Eds., Milan, July 2009, pp. 285 – 290, INSTICC.

[5] David W. Casbeer, Derek B. Kingston, Randal W. Beard, and Timothy W. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[6] Anand S. Rao and Michael P. Georgeff, "BDI Agents: From Theory to Practise," in *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, US*, Berlin / Heidelberg, 1995, pp. 312–319, Springer Verlag.

[7] Hartmut Schmeck, "Organic Computing – A new vision for distributed embedded systems," in *Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, 2005, pp. 201–203.

[8] Alan G. Ganek and Thomas A. Corbi, "The dawning of the autonomic computing era," *IBM Systems Journal*, vol. 42, no. 1, pp. 5–18, 2003.

[9] Christian Müller-Schloer, Christoph von der Malsburg, and Rolf P. Würtz, "Organic Computing," *Informatik Spektrum*, vol. 27, no. 4, pp. 332–336, 2004.

[10] David Tennenhouse, "Proactive computing," *Communications of the ACM*, vol. 43, no. 5, pp. 43–50, 2000.

[11] Christian Müller-Schloer and Hartmut Schmeck, "Organic Computing - Quo Vadis?," in *Organic Computing - A Paradigm Shift for Complex Systems*, Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, Eds., chapter 6.2, pp. 615 – 625. Birkhäuser Verlag, 2011.

[12] William Wright, David Schroh, Pascale Proulx, Alex Skaburskis, and Brian Cort, "The Sandbox for Analysis: Concepts and Methods," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, New York, NY, USA, 2006, CHI '06, pp. 801 – 810, ACM.

[13] Bennet Yee, David Sehr, Gregory Dardyk, J. Bradley Chen, Robert Muth, Tavis Ormandy, Shiki Okasaka, Neha Narula, and Nicholas Fullagar, "Native Client: A Sandbox for Portable, Untrusted x86 Native Code," in *Proceedings of the IEEE Symposium on Security and Privacy*, Los Alamitos, CA, USA, 2009, pp. 79 – 93, IEEE Computer Society.

[14] Katarzyna Keahey, Karl Doering, and Ian Foster, "From Sandbox to Playground: Dynamic Virtual Environments in the Grid," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Washington, DC, USA, 2004, GRID'04, pp. 34 – 42, IEEE Computer Society.

[15] Sven Tomforde, Holger Prothmann, Jürgen Branke, Jörg Hähner, Moez Mnif, Christian Müller-Schloer, Urban Richter, and Hartmut Schmeck, "Observation and Control of Organic Systems," in *Organic Computing - A Paradigm Shift for Complex Systems*, Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, Eds., chapter Chapter 4.1, pp. 325–338. Birkhäuser Verlag, 2011.

[16] Hartmut Schmeck, Christian Müller-Schloer, Emre Çakar, Moez Mnif, and Urban Richter, "Adaptivity and self-organization in organic computing systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 5, no. 3, pp. 1–32, 2010.

[17] Christian Müller-Schloer, "Organic Computing: On the feasibility of controlled emergence," in *CODES and ISSS 2004 Proceedings, September 8-10, 2004*. 2004, pp. 2–5, ACM Press.

[18] Abdelmajid Bouajila, Johannes Zeppenfeld, Walter Stechele, Andreas Herkersdorf, Andreas Bernauer, Oliver Bringmann, and Wolfgang Rosenstiel, "Organic Computing at the System on Chip Level," in *Proceedings of the IFIP International Conference on Very Large Scale Integration of System on Chip (VLSI-SoC)*, Berlin / Heidelberg, DE, October 2006, Springer Verlag.

[19] Andreas Bernauer, Johannes Zeppenfeld, Oliver Bringmann, Andreas Herkersdorf, and Wolfgang Rosenstiel, "Combining software and hardware LCS for lightweight on-chip learning," in *Proceedings of the 3rd IFIP Conference on Biologically-Inspired Collaborative Computing (BICC 2010)*, Berlin / Heidelberg, September 2010, pp. 279–290, Springer Verlag.

[20] Willi Richert, Ulrich Scheller, Markus Koch, Bernd Kleinjohann, and Claudius Stern, "Integrating sporadic imitation in reinforcement learning robots," in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL'09)*, Berlin / Heidelberg, DE, 2009, Springer Verlag.

[21] Adam El Sayed Auf, Marek Litza, and Erik Maehle, "Distributed fault-tolerant robot control architecture based on organic computing principles," in *Proceedings of the International Conference on Biologically-Inspired Collaborative Computing*, Berlin / Heidelberg, DE, 2008, pp. 115–124, Springer Verlag.

[22] Stefan Wildermann, Andreas Oetken, Jürgen Teich, and Zoran Salcic, "Self-organizing computer vision for robust object tracking in smart cameras," in *Autonomic and Trusted Computing*, Bing Xie, Juergen Branke, S. Sadjadi, Daqing Zhang, and Xingshe Zhou, Eds., vol. 6407 of *Lecture Notes in Computer Science*, pp. 1–16. Springer Berlin / Heidelberg, 2010.

[23] Rolf P. Würtz, Ed., *Organic Computing (Understanding Complex Systems)*, Springer Verlag, Berlin, DE, 2008.

[24] Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, Eds., *Organic Computing - A Paradigm Shift for Complex Systems*, Autonomic Systems. Birkhäuser Verlag, 2011.

[25] Jürgen Branke, Moez Mnif, Christian Müller-Schloer, Holger Prothmann, Urban Richter, Fabian Rochner, and Hartmut Schmeck, "Organic Computing – Addressing complexity by controlled self-organization," in *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, Tiziana Margaria, Anna Philippou, and Bernhard Steffen, Eds., 2006, pp. 200–206.

[26] Urban Richter, Moez Mnif, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck, "Towards a Generic Observer/Controller Architecture for Organic Computing," in *Beiträge zur Jahrestagung der Gesellschaft für Informatik 2006*, 2006, pp. 112–119.

[27] Moez Mnif and Christian Müller-Schloer, "Quantitative emergence," in *Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems (SMCals 2006)*, Piscataway, NJ, USA, July 2006, pp. 78–84, IEEE.

[28] Moez Mnif, *Quantitative Emergenz: Eine Quantifizierungsmethodik für die Entstehung von Ordnung in selbstorganisierenden technischen Systemen*, Ph.D. thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture Group, Hannover, DE, 2010.

[29] Urban Maximilian Richter, *Controlled Self-Organisation Using Learning Classifier Systems*, Ph.D. thesis, Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften, Karlsruhe, DE, July 2009.

[30] Emre Cakar, Jörg Hähner, and Christian Müller-Schloer, "Investigation of Generic Observer/Controller Architectures in a Traffic Scenario," in *INFORMATIK 2008: Beherrschbare Systeme – dank Informatik*, Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, and Christian Scheideler, Eds. 2008, vol. 134 of *Lecture Notes in Computer Science*, pp. 733–738, Köllen Verlag.

[31] Emre Cakar, Moez Mnif, Christian Müller-Schloer, Urban Richter, and Hartmut Schmeck, "Towards a quantitative notion of self-organisation," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore*. 2007, pp. 4222–4229, IEEE.

[32] Oliver Ribock, Urban Richter, and Hartmut Schmeck, "Using organic computing to control bunching effects," in *Proceedings of the 21th International Conference on Architecture of Computing Systems (ARCS 2008)*. Februar 2008, vol. 4934 of *LNCS*, pp. 232–244, Springer.

[33] Lutfi R. Al-Sharif, "Bunching in lifts: Why does bunching in lifts increase waiting time?," *Elevator World*, vol. 11, pp. 75–77, 1996.

[34] Micaela Wünsche, Sanaz Mostaghim, Hartmut Schmeck, Timo Kautzmann, and Marcus Geimer, "Organic Computing in Off-highway Machines," in *Second International Workshop on Self-Organizing Architectures*. The International Conference on Autonomic Computing and Communications, Juni 2010, pp. 51–58, ACM.

[35] Daniel Pathmaperuma, "Lernende und selbstorganisierende Putzroboter," Master thesis, Institut AIFB, Univ. Karlsruhe (TH), Karlsruhe, DE, 2008.

[36] Clemens Lode, Urban Richter, and Hartmut Schmeck, "Adaption of XCS to Multi-Learner Predator/Prey Scenarios," in *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010)*. 2010, pp. 1015–1022, ACM.

[37] Stefan Thanheiser, Lei Liu, and Hartmut Schmeck, "Towards Collaborative Coping with IT Complexity by Combining SOA and Organic Computing," *International Transactions on Systems Science and Applications*, vol. 5, no. 2, pp. 190–197, September 2009.

[38] Lei Liu, Stefan Thanheiser, and Hartmut Schmeck, "A reference architecture for self-organizing service-oriented computing," in *Architecture of Computing Systems - ARCS 2008, 21st International Conference, Dresden, Germany, February 25-28, 2008, Proceedings*, Uwe Brinkschulte, Theo Ungerer, Christian Hochberger, and Rainer G. Spallek, Eds., Berlin / Heidelberg, 2008, vol. 4934 of *Lecture Notes in Computer Science*, pp. 205–219, Springer Verlag.

[39] Lei Liu and Hartmut Schmeck, "Enabling self-organising service level management with automated negotiation," *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 42–45, 2010.

[40] Birger Becker, Florian Allerding, Ulrich Reiner, Mattias Kahl, Urban Richter, Daniel Pathmaperuma, Hartmut Schmeck, and Thomas Leibfried, "Decentralized energy-management to control smart-home architectures," in *Architecture of Computing Systems - ARCS 2010, 23rd International Conference, Hannover, Germany, February 22-25, 2010. Proceedings*, Christian Müller-Schloer, Wolfgang Karl, and Sami Yehia, Eds., Berlin / Heidelberg, DE, 2010, vol. 5974 of *Lecture Notes in Computer Science*, pp. 150–161, Springer Verlag.

[41] Web, "The OCCS Website," http://projects.aifb.kit.edu/effalg/otcqe/qe/index.htm, Universität Karlsruhe (AIFB) and Leibniz Universität Hannover (ISE-SRA), 2010.

[42] Rodney A. Brooks, "How to build complete creatures rather than isolated cognitive simulators," in *Architectures for Intelligence*. 1991, pp. 225–239, Erlbaum Publishers.

[43] Oliver Louis Robert Jacobs, *Introduction to Control Theory*, Oxford University Press, Oxford, UK, 2nd edition, 1993.

[44] Karl Johan Astrom and Roichard M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, Princeton University Press, Princeton, NJ, US, 2008.

[45] Karl J. Astrom and Bjorn Wittenmark, *Adaptive Control*, Dover Publications Inc., Mineola, NY, US, 2nd edition, 2008.

[46] Stuart Bennett, *A History of Control Engineering 1930-1955*, Peter Peregrinus, Hitchin, UK, 1st edition, 1993.

[47] Kiam Heong Ang, Gregory Chong, and Yun Li, "PID Control System Analysis, Design, and Technology," *IEEE transactions on control system technology*, vol. 13, no. 4, pp. 559–576, 2005.

[48] Eduardo F. Camacho and Carlos Bordons, *Model Predictive Control*, Springer Verlag, Berlin / Heidelberg, DE, 2004.

[49] Carlos E. Garcia, David M. Prett, and Manfred Morari, "Model predictive control: theory and practice – a survey," *Automatica*, vol. 25, pp. 335–348, May 1989.

[50] David Q. Mayne, James B. Rawlings, Christopher V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[51] Moritz Diehl, H.Georg Bock, Johannes P. Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer, "Real-time optimisation and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[52] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini, *Feedback Control of Dynamic Systems*, Prentice Hall, Upper Saddle River, NJ, US, 5th edition, 2005.

[53] Robert H. Cannon, *Dynamics of Physical Systems*, Dover Publications Inc., Mineola, NY, US, 2003.

[54] Vladimir I. Arnold, *Mathematical Methods of Classical Mechanics*, Graduate Texts in Mathematics. Springer Verlag, Berlin, DE, 2nd edition, 1989.

[55] Prabha Kundur, *Power System Stability and Control*, Epri Power System Engineering Series. Mcgraw-Hill Professional, New York, US, 1994.

[56] John Watton, *Fundamentals of Fluid Power Control*, Cambridge University Press, Cambridge, UK, 1st edition, 2009.

[57] Bruno Siciliano and Luigi Villani Lorenzo Sciavicco and, *Robotics: Modelling, Planning and Control*, Advanced Textbooks in Control and Signal Processing. Springer Verlag, Berlin / Heidelberg, DE, 1st edition, 2008.

[58] Howie Choset, Seth Hutchinson, and George Kantor, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, Intelligent Robotics and Autonomous Agents. The MIT Press, Cambridge, MA, US, 2005.

[59] Joachim Hertzberg, Herbert Jaeger, Uwe Zimmer, and Philippe Morignot, "A framework for plan execution in behavior-based robots," in *Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings*, Sept. 1998, pp. 8 –13.

[60] Rodney A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14 – 23, March 1986.

[61] Erann Gat, "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots," in *Proceedings of the tenth national conference on Artificial intelligence*. 1992, AAAI'92, pp. 809–815, AAAI Press.

[62] Murray Shanahan and Mark Witkowski, "High-level robot control through logic," in *Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*, London, UK, 2001, ATAL '00, pp. 104–121, Springer-Verlag.

[63] Anthony Joseph Stentz, *The Navlab system for mobile robot navigation*, Ph.D. thesis, Pittsburgh, PA, USA, 1990, UMI Order No. GAX90-33071.

[64] Rodney A. Brooks, "A robot that walks; emergent behaviors from a carefully evolved network," *Neural Computation*, vol. 1, pp. 253–262, June 1989.

[65] Jonathan Connell, "A colony architecture for an artificial creature," Tech. Rep., Cambridge, MA, USA, 1989.

[66] Rodney A. Brooks, Jonathan H. Connell, and Peter Ning, "Herbert: A second generation mobile robot," Tech. Rep., Cambridge, MA, USA, 1988.

[67] Bram Bakker, Viktor Zhumatiy, Gabriel Gruener, and Jürgen Schmidhuber, "Quasi-online Reinforcement Learning for Robots," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA'06)*, 2006.

[68] Srikanta Patnaik and Kuravateppa Karibasappa, "Cognition techniques and their applications," in *Intelligent Knowledge-Based Systems*, Cornelius T. Leondes, Ed., pp. 1018 – 1065. Springer US, New York, US, 2005.

[69] Michael J. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons Publishers, Hoboken, NJ, US, 2nd edition, 2009.

[70] Nicholas R. Jennings, "Building complex software systems: The case for an agent-based approach," *Communications of the ACM, Forthcoming*, vol. 44, pp. 12–23, 2000.

[71] Ron Sun and Isaac Naveh, "Simulating organizational decision-making using a cognitively realistic agent model," *Journal of Artificial Societies and Social Simulation*, vol. 7, no. 3, 2004.

[72] Nathan Schurr, Janusz Marecki, John P. Lewis, Milind Tambe, and Paul Scerri, "The defacto system: Coordinating human-agent teams for the future of disaster response," in *Multi-Agent Programming: Languages, Platforms and Applications*, Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, Eds., pp. 197–215. Springer Verlag, Berlin, DE, 2005.

[73] Alex Rogers, Esther David, Nicholas R. Jennings, and Jeremy Schiff, "The effects of proxy bidding and minimum bid increments within ebay auctions," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 2, pp. 9, 2007.

[74] Liviu Panait and Sean Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, pp. 387–434, November 2005.

[75] Stanley J. Rosenschein and Leslie Pack Kaelbling, "The synthesis of digital machines with provable epistemic properties," in *TARK '86: Proceedings of the 1986 conference on Theoretical aspects of reasoning about knowledge*, San Francisco, CA, US, 1986, pp. 83–98, Morgan Kaufmann Publishers Inc.

[76] Michael E. Bratman, David Israel, and Martha Pollack, "Plans and resource – bounded practical reasoning," in *Philosophy and AI: Essays at the Interface*, Robert Cummins and John L. Pollock, Eds., pp. 1–22. The MIT Press, Cambridge, Massachusetts, 1988.

[77] W. Brian Arthur, "Complexity in economic theory: Inductive reasoning and bounded rationality," *The American Economic Review*, vol. 84, no. 2, pp. 406–411, May 1994.

[78] Anand S. Rao and Michael P. Georgeff, "Modeling Rational Agents within a BDI-Architecture," in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, San Francisco, CA, US, 1991, pp. 201–203, Morgan Kaufmann publishers Inc.

[79] Yoav Shoham, "Agent-oriented programming," *Artificial Intelligence*, vol. 60, no. 1, pp. 51–92, 1993.

[80] Jon Doyle, "Rationality and its roles in reasoning," *Computational Intelligence*, vol. 8, pp. 376–409, 1994.

[81] Michael E. Bratman, *Intention, Plans, and Practical Reason*, Center for the Study of Language and Information (CSLI) Publications, Stanford, CA, US, 1999.

[82] David L. Martin, Adam Cheyer, and Douglas B. Moran, "The open agent architecture: A framework for building distributed software systems," *Applied Artificial Intelligence*, vol. 13, no. 1-2, pp. 91–128, 1999.

[83] Sanjeev Kumar, Philip R. Cohen, and Hector J. Levesque, "The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams," in *In Proceedings of the Fourth International Conference on Multi-Agent Systems*. 2000, pp. 159–166, IEEE Computer Society.

[84] The Cougaar Open Source Website, "http://www.cougaar.org," Web, 2009.

[85] Karl Kleinmann, Richard Lazarus, and Ray Tomlinson, "An Infrastructure for Adaptive Control of Multi-Agent Systems," in *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2003.

[86] Michael Jarrett and Rudolph Seviora, "Constructing an autonomic computing infrastructure using cougaar," in *Engineering of Autonomic and Autonomous Systems, 2006. EASe 2006. Proceedings of the Third IEEE International Workshop on*, March 2006, pp. 119–128.

[87] Gerhard Weiß, "Adaptation and Learning in Multi-Agent Systems: Some Remarks and a Bibliography," in *Proceedings of the Workshop on Adaption and Learning in Multi-Agent Systems*, London, UK, 1996, IJCAI '95, pp. 1–21, Springer-Verlag.

[88] Giovanna Di Marzo Serugendo, Marie Pierre Gleizes, and Anthony Karageorgos, "Self-Organisation and Emergence in MAS: An Overview," *Informatica (Slovenia)*, vol. 30, no. 1, pp. 45–54, 2006.

[89] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf, "Jadex: A BDI Reasoning Engine," in *Multi-Agent Programming: Languages, Platforms and Applications*, Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, Eds., vol. 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pp. 149–174. Springer Verlag, Berlin / Heidelberg, DE, 2005.

[90] Toan Phung, Michael Winikoff, and Lin Padgham, "Learning within the bdi framework: An empirical analysis," in *Knowledge-Based Intelligent Information and Engineering Systems - 9th International Conference, KES 2005, Melbourne, Australia, September 14-16, 2005, Proceedings, Part III*, Berlin / Heidelberg, 2005, vol. 3683/2005, pp. 282–288, Springer Verlag.

[91] Alejandro Guerra-Hernandez, Amal El Fallah-Seghrouchni, and Henry Soldano, "Learning in bdi multi-agent systems," in *Computational Logic in Multi-Agent Systems, 4th International Workshop, CLIMA IV, Fort Lauderdale, FL, USA, January 6-7, 2004, Revised Selected and Invited Papers*, Jürgen Dix and João Alexandre Leite, Eds., Berlin / Heidelberg, 2004, pp. 218–233, Springer Verlag.

[92] Michael P. Georgeff, Barney Pell, Martha E. Pollack, Milind Tambe, and Michael Wooldridge, "The belief-desire-intention model of agency," in *ATAL '98: Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, Berlin / Heidelberg, DE, 1999, pp. 1–10, Springer Verlag.

[93] Anand S. Rao and Michael P. Georgeff, "Formal models and decision procedures for multi-agent systems," Tech. Rep. 61, Australian Artificial Institute, June 1995.

[94] Jiming Liu, *Autonomous agents and multi-agent systems: explorations in learning, self-organization and adaptive computation*, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2001.

[95] IBM Corporation, "An architectural blueprint for autonomic computing," *IBM Whitepaper*, 2004.

[96] Steve R. White, James E. Hanson, Ian Whalley, David M. Chess, and Jeffrey O. Kephart, "An Architectural Approach to Autonomic Computing," in *Proceedings of the 1st International Conference on Autonomic Computing (ICAC 2004)*, 2004, pp. 2–9.

[97] Jana Koehler, Chris Giblin, Dieter Gantenbein, and Rainer Hauser, "On autonomic computing architectures," Tech. Rep., IBM Zurich Research Laboratory, 2003.

[98] M. Muztaba Fuad and Michael J. Oudshoorn, "System architecture of an autonomic element," in *Proceedings of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems (EASE'07)*, Washington, DC, USA, 2007, pp. 89–93, IEEE Computer Society.

[99] Yu Cheng, Ramy Farha, Myung Sup Kim, Alberto Leon-Garcia, and James Won-Ki Hong, "A generic architecture for autonomic service and network management," *Computer Communications*, vol. 29, no. 18, pp. 3691–3709, 2006.

[100] Yan Zhang, Anna Liu, and Wei Qu, "Software architecture design of an autonomic system," in *Proceedings of 5th Australasian Workshop on Software and System Architectures*, 2004, pp. 5–11.

[101] Manish Agarwal, Viraj Bhat, Hua Liu, Vincent Matossian, Vladimir Putty, Cristina Schmidt, Guangsen Zhang, Liang Zhen, Manish Parashar, Bithika Khargharia, and Salim Hariri, "AutoMate: Enabling Autonomic Applications on the Grid," in *Proceedings of the 5th Annual International Workshop on Active Middleware Services (AMS 2003), 25 June 2003, Seattle, WA, USA*, 2003, pp. 48 – 59.

[102] Hua Liu and Manish Parashar, "Accord: A Programming Framework For Autonomic Applications," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 36, no. 3, pp. 341–352, May 2006.

[103] David M. Chess, Alla Segal, Ian Whalley, and Steve R. White, "Unity: experiences with a prototype autonomic computing system," in *Proceedings of the International Conference on Autonomic Computing (ICAC'04)*, May 2004, pp. 140–147.

[104] Radu Calinescu, "Resource-definition policies for autonomic computing," in *Proceedings of the 5th International Conference on Autonomic and Autonomous Systems (ICAS'09)*. April 2009, pp. 111 – 116, IEEE Computer Society Press.

[105] Radu Calinescu, "Model-driven autonomic architecture," in *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*, Washington, DC, USA, 2007, p. 9, IEEE Computer Society.

[106] Radu Calinescu, "Implementation of a generic autonomic framework," in *ICAS '08: Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems*, Washington, DC, USA, 2008, pp. 124–129, IEEE Computer Society.

[107] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli, "A survey of autonomic communications," *ACM Transactions on Autonomous Adaptaptive Systems*, vol. 1, no. 2, pp. 223–259, 2006.

[108] Uwe Hansmann, Lothar Merk, Martin S. Nicklous, and Thomas Stober, *Pervasive Computing : The Mobile World*, Springer Verlag, Berlin / Heidelberg, DE, August 2003.

[109] Mark Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, September 1991.

[110] Jeff Kramer and Jeff Magee, "Self-Managed Systems: an Architectural Challenge," *Future of Software Engineering*, pp. 259 – 268, 2007.

[111] Thorsten Hestermeyer, Oliver Oberschelp, and Holger Giese, "Structured Information Processing For Self-optimizing Mechatronic Systems," in *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004), Setubal, Portugal*, Helder Araujo, Alves Vieira, Jose Braz, Bruno Encarnacao, and Marina Carvalho, Eds. 8 2004, pp. 230–237, INSTICC Press.

[112] John J. Grefenstette and Connie L. Ramsey, "An Approach to Anytime Learning," in *Proceedings of the 9th International Workshop on Machine Learning*, 1992, pp. 189–195.

[113] Sven Burmester, Holger Giese, Eckehard Münch, Oliver Oberschelp, Florian Klein, and Peter Scheideler, "Tool support for the design of self-optimizing mechatronic multi-agent systems," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 10, pp. 207–222, 2008.

[114] John J. Grefenstette, Connie Loggia Ramsey, and Alan C. Schultz, "Learning sequential decision rules using simulation models and competition," *Machine Learning*, vol. 5, pp. 355–381, 1990.

[115] Helen G. Cobb and John J. Grefenstette, "Learning the persistence of actions in reactive control rules," in *Proceedings of the Eighth International Machine Learning Workshop*, Evanston, Il, US, 1991, pp. 292–297, Morgan Kaufmann Publishers Inc.

[116] Thomas Bäck and Hans-Paul Schwefel, "Evolutionary Computing: An Overview," in *Proceedings of IEEE Conference of Evolutionary Computing*, 1996, pp. 20 – 29.

[117] Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intelligent Systems*, vol. 14, pp. 54–62, May 1999.

[118] Marco Dorigo, "ALECSYS and the AutonoMouse: Learning to Control a Real Robot by Distributed Classifier Systems," *Machine Learning*, vol. 19, pp. 209–240, June 1995.

[119] Jean-Yves Donnart and Jean-Arcady Meyer, "Learning reactive and planning rules in a motivationally autonomous animat," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 3, pp. 381–395, June 1996.

[120] Hartmut Schmeck and Christian Müller-Schloer, "A characterization of key properties of environment-mediated multiagent systems," in *Engineering Environment-Mediated Multi-Agent Systems, International Workshop, EEMMAS 2007, Dresden, Germany, October 5, 2007*, Danny Weyns, Sven A. Brueckner, and Yves Demazeau, Eds. 2007, pp. 17–38, Springer Verlag, Berlin, Heidelberg.

[121] Sven Tomforde and Jörg Hähner, *Biologically Inspired Networking and Sensing: Algorithms and Architectures*, chapter Organic Network Control – Turning standard protocols into evolving systems, p. to appear, IGI, 2011.

[122] David E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, MA, US, 1989.

[123] Stafford Beer, *Diagnosing the System for Organizations*, Managerial Cybernetics of Organization. John Wiley & Sons Publishers, Hoboken, NJ, US, 1994.

[124] George Liu and Gerald Maguire, Jr., "A class of mobile motion prediction algorithms for wireless mobile computing and communication," *Mobile Networks and Applications; Special issue: routing in mobile communications networks*, vol. 1, no. 2, pp. 113–121, 1996.

[125] Tom M. Mitchell, *Machine Learning*, Computer Science Series. McGraw-Hill Companies, Inc., Singapure, 1997.

[126] Zbigniew Michalewicz and David B. Fogel, *How to Solve It: Modern Heuristics*, Springer Verlag, Berlin / Heidelberg, 2004, ISBN: 3540224947.

[127] Michael J. North, Nicholson T. Collier, and Jerry R. Vos, "Experiences creating three implementations of the repast agent modeling toolkit," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 16, no. 1, pp. 1–25, 2006.

[128] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, and Keith Sullivan, "MASON: A New Multi-Agent Simulation Toolkit," in *Proceedings of the 2004 Swarmfest Workshop*, 2004.

[129] 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*, vol. 41, no. 2–3, pp. 173–203, 2005.

[130] Kevin Fall, "Network Emulation in the Vint/NS Simulator," in *Proceedings of the The Fourth IEEE Symposium on Computers and Communications (ISCC'99)*, Washington, DC, USA, 1999, p. 244, IEEE Computer Society.

[131] Andras Varga, "The OMNET++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference*, Prague, Czech Republic, June 2001, pp. 319–324, SCS – European Publishing House.

[132] National Electrical Manufacturers Association, "NEMA Standards Publication TS 2-2003 v02.06 – Traffic Controller Assemblies with NTCIP Requirements," 2003.

[133] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, US, 1998.

[134] Björn Hurling, "Bewertung und Implementierung von verschiedenen Lernverfahren zur automatischen Optimierung von Netzwerkprotokollparametern," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, August 2009.

[135] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition, 2010.

[136] Ethem Alpaydim, *Maschinelles Lernen*, Oldenbourg Wissenschaftsverlag GmbH, München, 2008.

[137] Charles W. Anderson, "Learning to Control an Inverted Pendulum Using Neural Networks," *IEEE Control Systems Magazine*, vol. 9, no. 3, pp. 31–37, 1989.

[138] Jorge Casillas, Brian Carse, and Larry Bull, "Fuzzy-XCS: A Michigan Genetic Fuzzy System," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 4, pp. 536 – 550, 2007.

[139] Stewart W. Wilson, "ZCS: A zeroth level classifier system," *Evolutionary Computation*, vol. 2, no. 1, pp. 1–18, 1994.

[140] Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang, "Readings in speech recognition," chapter Phoneme recognition using time-delay neural networks, pp. 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[141] Dean A. Pomerleau, "Advances in neural information processing systems 1," chapter ALVINN: an autonomous land vehicle in a neural network, pp. 305–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[142] Michael J. Kurtz, Piero Mussio, and Peter G. Ossorio, "A cognitive system for astronomical image interpretation," *Pattern Recognition Letters*, vol. 11, no. 7, pp. 507 – 515, 1990.

[143] Gerald Tesauro, "Temporal difference learning and TD-Gammon," *Communications of the ACM*, vol. 38, pp. 58 – 68, March 1995.

[144] Igor V. Tetko, David J. Livingstone, and Alexander I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[145] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541 – 551, December 1989.

[146] Jerome Bruner, Jacqueline Jarrett Goodnow, and George Austin, *A Study of Thinking*, Social Science Classics Series. Transaction Publishers, 1986.

[147] Richard E. Neapolitan, *Learning Bayesian Networks*, Prentice Hall, Upper Saddle River, NJ, USA, 2003.

[148] Kevin Gurney, *An Introduction to Neural Networks*, CRC Press, 1997.

[149] Christopher John Cornish Hellaby Watkins, *Learning from Delayed Rewards*, Ph.D. thesis, Kings College, University of Cambridge, Cambridge, GB, May 1989.

[150] Arthur L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210 – 229, July 1959.

[151] Richard S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine*, vol. 3, pp. 9 – 44, August 1988.

[152] John H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA, 1975.

[153] John H. Holland and Judith S. Reitman, "Cognitive systems based on adaptive algorithms," *ACM SIGART Bulletin*, p. 49, June 1977.

[154] Tim Kovacs and Pier Luca Lanzi, "A bigger learning classifier systems bibliography," in *Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems*, London, UK, 2001, IWLCS '00, pp. 213–252, Springer-Verlag.

[155] John H. Holland, Lashon B. Booker, Marco Colombetti, Marco Dorigo, David E. Goldberg, Stephanie Forrest, Rick L. Riolo, Robert E. Smith, Pier Luca Lanzi, Wolfgang Stolzmann, and Stewart W. Wilson, "What is a learning classifier system?," in *Learning Classifier Systems, From Foundations to Applications*, London, UK, 2000, pp. 3–32, Springer-Verlag.

[156] Jorge Casillas, Brian Carse, and Larry Bull, "Fuzzy XCS: An Accuracy-Based Fuzzy Classiffier System," in *Proceedings of the XII Congreso Espanol sobre Tecnologia y Logica Fuzzy (ESTYLF 2004)*, 2004.

[157] Stephen Frederick Smith, *A learning system based on genetic adaptive algorithms*, Ph.D. thesis, Pittsburgh, PA, USA, 1980, AAI8112638.

[158] Stephen F. Smith, "Flexible learning of problem solving heuristics through adaptive search," in *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 1*, San Francisco, CA, USA, 1983, pp. 422–425, Morgan Kaufmann Publishers Inc.

[159] Stewart W. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[160] Martin V. Butz, "XCSJava 1.0: An Implementation of the XCS classifier system in Java," Technical Report 2000027, Illinois Genetic Algorithms Laboratory, 2000.

[161] Martin Butz and Stewart W. Wilson, "An algorithmic description of xcs," in *IWLCS '00: Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems*, London, UK, 2000, pp. 253–272, Springer-Verlag.

[162] Martin V. Butz and Steward W. Wilson, "An algorithmic description of XCS," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 6, pp. 144–153, 2002.

[163] Fabian Rochner and Christian Müller-Schloer, "Adaptive decentralized and collaborative control of traffic lights," in *INFORMATIK 2004 - Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004*, Peter Dadam and Manfred Reichert, Eds. 2004, vol. 2, pp. 595–599, GI.

[164] Fabian Rochner, Holger Prothmann, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck, "An organic architecture for traffic light controllers," in *Informatik 2006 – Informatik für Menschen*, Christian Hochberger and Rüdiger Liskowsky, Eds. 2006, vol. P-93 of *LNI*, pp. 120–127, Köllen Verlag.

[165] Christian Becker, "Untersuchung von Mechanismen zur technischen Umsetzbarkeit von Flexibilitaet in Organischen Systemen," Diploma thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, May 2011.

[166] John H. Holland, "Computation & intelligence," chapter Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems, pp. 275–304. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1995.

[167] Stewart Wilson, "Get Real! XCS with Continuous-Valued Inputs," in *Learning Classifier Systems*, Pier Lanzi, Wolfgang Stolzmann, and Stewart Wilson, Eds., vol. 1813 of *Lecture Notes in Computer Science*, pp. 209–219. Springer Berlin / Heidelberg, 2000.

[168] Manuel Valenzuela-Rendón, "The fuzzy classifier system: A classifier system for continuously varying variables," in *Proceedings of the 4th International Conference on Genetic Algorithms, San Diego, CA, USA, July 1991*, Richard K. Belew and Lashon B. Booker, Eds. 1991, pp. 346–353, Morgan Kaufmann.

[169] Lotfi A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[170] Andrea Bonarini, "Learning to coordinate fuzzy behaviors for autonomous agents," *International Journal of Approximate Reasoning*, vol. 17, no. 4, pp. 409 – 432, 1994.

[171] Charles E. Perkins and Pravin Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.

[172] Jiang Y. Ke, Kai S. Tang, and Ken F. Man, "Genetic Fuzzy Classifier for Benchmark Cancer Diagnosis," in *Proceedings of 23rd International Conference on Industrial Electronics, Control and Instrumentation (IECON97)*, 1997, vol. 3, pp. 1063 – 1067.

[173] George J. Klir and Bo Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, 2nd edition edition, 1995.

[174] Bram Cohen, "Incentives Build Robustness in BitTorrent," in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley*, 2003.

[175] Kolja Eger, Tobias Hossfeld, Andreas Binzenhöfer, and Gerald Kunzmann, "Efficient Simulation of Large-Scale P2P Networks: Packet-level vs. Flow-level Simulations," in *Proceedings of the 2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN'07)*, Monterey Bay, USA, 2007, pp. 9–16.

[176] David H. Wolpert and William G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67 – 82, 1997.

[177] James Kennedy, Russell C. Eberhart, and Yuhui Shi, *Swarm Intelligence*, Morgan Kaufmann Series in Evolutionary Computation. Morgan Kaufmann, San Francisco, CA, US, 2001.

[178] Thomas Weise, "Global optimization algorithms - theory and application," http://www.it-weise.de/, December 2010.

[179] Karin Zielinski, *Optimizing Real-World Problems with Differential Evolution and Particle Swarm Optimization*, Ph.D. thesis, Universität Bremen, Shaker Verlag Aachen, DE, 2009.

[180] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, US, 2nd edition edition, 2001.

[181] Ivo Nowak, *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*, International Series of Numerical Mathematics. Birkhäuser Verlag, Basel, CH, 2005.

[182] Agoston E. Eiben and James E. Smith, *Introduction to Evolutionary Computing*, Springer Verlag, 2nd edition, 2007.

[183] Jakob Vesterstrom, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the IEEE Congress on Evolutionary Computation.* 2004, vol. 2, pp. 1980–1987, IEEE.

[184] Dinkar N. Bhat, "An evolutionary measure for image matching," in *Proceedings of the 14th International Conference on Pattern Recognition-Volume 1 - Volume 1*, Washington, DC, USA, 1998, ICPR '98, pp. 850–, IEEE Computer Society.

[185] Rainer Storn and Kenneth Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[186] Rasmus K. Ursem and Pierre Vadstrup, "Parameter identification of induction motors using differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*. 2003, vol. 2, pp. 790–796, IEEE.

[187] Russell C. Eberhart and James Kennedy, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[188] Jovita Nenortaite and Rimantas Butleris, "Application of particle swarm optimization algorithm to decision making model incorporating cluster analysis," in *Proceedings of the 2008 Conference on Human System Interactions*. May 2008, pp. 88–93, IEEE.

[189] Xin-She Yang, "Harmony search as a metaheuristic algorithm," in *Music-Inspired Harmony Search Algorithm*, Zong Geem, Ed., vol. 191 of *Studies in Computational Intelligence*, pp. 1–14. Springer Verlag, Berlin / Heidelberg, DE, 2009.

[190] Kang S. Lee and Zong W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36-38, pp. 3902–3933, September 2005.

[191] Aditya Panchal, "Harmony search in therapeutic medical physics," in *Music-Inspired Harmony Search Algorithm*, Zong Geem, Ed., vol. 191 of *Studies in Computational Intelligence*, pp. 189–203. Springer Berlin / Heidelberg, 2009.

[192] Scott Kirkpatrick, David Gelatt, and Mario P. Vecchi, "Optimization by Simulated Annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

[193] Horacio Martinez-Alfaro and Donald Flugrad, "Collision-free path planning for mobile robots and/or agvs using simulated annealing," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, October 1994, vol. 1, pp. 270–275.

[194] Markus Domdey, "Vergleich von Optimierungsverfahren für organische Systeme am Beispiel von Organic Network Control," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, April 2010.

[195] Colin R. Reeves, "Fitness landscapes," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Edmund K. Burke and Graham Kendall, Eds., chapter 19, pp. 587–610. Springer Verlag, Berlin / Heidelberg, DE, 2005.

[196] Stuart A. Kauffman, *The Origins of Order: Self-Organisation and Selection in Evolution*, Oxford University Press, 1993.

[197] Jani Rönkkönen, Saku Kukkonen, and Kenneth Price, "Real-parameter optimization with differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*. IEEE, 2005, pp. 506–513.

[198] Anthony Carlisle and Gerry Dozier, "An off-the-shelf PSO," in *Proceedings of the workshop on particle swarm optimization*, Indianapolis, IN, US, 1987, Purdue School of Engineering and Technology.

[199] Ihor O. Bohachevsky, Mark E. Johnson, and Myron L. Stein, "Generalized simulated annealing for function optimization," *Technometrics*, vol. 28, pp. 209–217, August 1986.

[200] Dennis I. Robertson and R. David Bretherton, "Optimizing networks of traffic signals in real time – the SCOOT method," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 11–15, 1991.

[201] Arthur G. Sims and Kenneth W. Dobinson, "The Sydney coordinated adaptive traffic (SCAT) system – Philosophy and benefits," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130–137, 1980.

[202] Peter Dürr, *Integration des ÖPNV in die dynamische Fahrwegsteuerung des Strassenverkehrs – Steuerungsverfahren Darvin*, Veröffentlichung des Lehrstuhls für Verkehrs- und Stadtplanung, Technische Universität München, 2001.

[203] Heinz Zackor, Fritz Busch, and Winfried Höpfl, "Entwicklung eines Verfahrens zur adaptiven koordinierten Steuerung von Lichtsignalanlagen," in *Forschung Straßenbau und Straßenverkehrstechnik*, Bonn-Bad Godesberg, Dec. 1990, vol. 607 / 1991, Bundesminister für Verkehr.

[204] Andy Tomlinson and Larry Bull, "An investigation into LCS road traffic junction controllers," Tech. Rep., Faculty of Computing, Engineering & Mathematical Sciences, Bristol, 2001.

[205] Yanan J. Cao, Neil Ireson, Larry Bull, and Ray Miles, "Design of a Traffic Junction Controller Using Classifier System and Fuzzy Logic," in *Computational Intelligence – Theory and Applications*, Bernd Reusch, Ed. 1999, vol. 1625 of *LNCS*, pp. 342–353, Springer.

[206] Yanan J. Cao, Neil Ireson, Larry Bull, and Ray Miles, "Distributed Learning Control of Traffic Signals," in *Real-World Applications of Evolutionary Computing – EvoWorkshops Proceedings*, Stefano Cagnoni et al., Eds. 2000, vol. 1803 of *LNCS*, pp. 117–126, Springer.

[207] Hartmut Schmeck, "Optimierungstechniken des Organic Computing in der Verkehrstechnik," in *Informatik bewegt! Informationstechnik in Verkehr und Logistik*, Andreas Pfingsten and Franz Rammig, Eds., pp. 11–38. Fraunhofer-IRB-Verlag, 2007.

[208] Holger Prothmann, Sven Tomforde, Jürgen Branke, Jörg Hähner, Christian Müller-Schloer, and Hartmut Schmeck, "Organic Traffic Control," in *Organic Computing – A Paradigm Shift for Complex Systems*, Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, Eds., chapter Chapter 5.1, pp. 431–446. Birkhäuser Verlag, 2011.

[209] Swiss Verkehrs-Systeme AG, "VS-Plus webpage," Online, `http://www.vs-plus.de`, 2008.

[210] Joachim Mertz, *Ein mikroskopisches Verfahren zur verkehrsadaptiven Knotenpunktsteuerung mit Vorrang des öffentlichen Verkehrs*, Dissertation, Fachgebiet Verkehrstechnik und Verkehrsplanung der Technischen Universität München, München, 2001.

[211] Christina Diakaki, *Integrated control of traffic flow in corridor networks*, Ph.D. thesis, Technical University of Crete, Department of Production Engineering and Management, Chania, Greece, 1999.

[212] Christina Diakaki, Vaya Dinopoulou, Kostas Aboudolas, Markos Papageorgiou, Elia Ben-Shabat, Eran Seider, and Amit Leibov, "Extensions and new applications of the traffic signal control strategy TUC," in *Transportation Research Record No. 1856*. 2003, pp. 202–211, Transportation Research Board.

[213] Vaya Dinopoulou, Christina Diakaki, and Markos Papageorgiou, "Applications of the urban traffic control strategy tuc," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1652–1665, 2006.

[214] Bernhard Friedrich, *Ein verkehrsadaptives Verfahren zur Steuerung von Lichtsignalanlagen*, Veröffentlichung des Fachgebiets Verkehrstechnik und Verkehrsplanung. Technische Universität München, 1999.

[215] Bernhard Friedrich, "Steuerung von Lichtsignalanlagen, BALANCE - ein neuer Ansatz," *Strassenverkehrstechnik, Kirschbaum Verlag, Bonn, DE*, vol. 7, 2000.

[216] Bernhard Friedrich, "Models for Adaptive Urban Traffic Network Control," in *Proceedings of the 8th Meeting of the Euro Working Group Transportation*, Rom, 2000.

[217] Christopher Toomey, Bernhard Friedrich, and Michael Clark, "BALANCE – a European field trial," in *Proceedings of the 9th International Conference on Road Transport Information and Control, 1998*, 1998, number Conference Publication No. 454, pp. 95 – 99.

[218] N. H. Gartner, "Demand-responsive decentralised urban traffic control," Tech. Rep. DOT/RSPA/DPB-50/81/24, US Department of Transportation, 1982.

[219] N. H. Gartner, "OPAC Strategy for demand-responsive decentralized traffic signal control," in *Control, Computers, Communications in Transportation*, J.-P. Perrin, Ed., 1989.

[220] Florence Boillot, "Optimal Signal Control of Urban Traffic Networks," in *Proceedings of the 6th International Conference on Road Traffic Monitoring and Control*, 1992, pp. 75–79.

[221] 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*, vol. 14, no. 1, pp. 18 – 38, 2006.

[222] F. Donati, V. Mauro, G. Roncolini, and M. Vallauri, "A Hierarchical Decentralised Traffic Light Control System. The First Realisation: 'Progetto Torino'," in *Proceedings of the 9th IFAC World Congress*, 1984, vol. II, 11G/A-1.

[223] Jean F. Barriere, Jean-Loup Farges, and Jean-Jacques Henry, "Decentralization vs hierarchy in optimal traffic control," in *Proceedings of the 5th IFAC/IFIP/IFORS International Conference on Control in Transportation Systems*, 1986, pp. 209–214.

[224] Jean-Jacques Henry, Jean-Loup Farges, and Jean Tuffal, "The PRODYN real time traffic algorithm," in *Proceedings of the 4th IFAC-IFIF-IFORS Conference on Control in Transportation Systems*, Baden-Baden, DE, 1983.

[225] Jean-Jacques Henry and Jean-Loup Farges, "PRODYN," in *Control, Computers, Communications in Transportation*, J.-P. Perrin, Ed., 1989.

[226] Jean-Loup Farges, Louahdi Khoudour, and Jean B. Lesort, "Prodyn: on site evaluation," in *Proceedings of the Third International Conference on Road Traffic Control, 1990*, 1-3 1990, pp. 62–66.

[227] Dirk Helbing, Stefan Lämmer, and Jean-Patrick Lebacque, "Self-organized control of irregular or perturbed network traffic," in *Optimal Control and Dynamic Games*, Christophe Deissenberg and Richard F. Hartl, Eds., pp. 239–274. Springer, Dordrecht, 2005.

[228] Martin Treiber and Dirk Helbing, "Visualisierung der fahrzeugbezogenen und verkehrlichen Dynamik mit und ohne Beeinflussungs-Systemen," in *Simulation und Visualisierung 2004 (SimVis 2004) 4-5 März 2004, Magdeburg*, Thomas Schulze, Stefan Schlechtweg, and Volkmar Hinz, Eds. 2004, pp. 323–334, SCS Publishing House e.V.

[229] Stefan Lämmer, *Reglerentwurf zur dezentralen Online-Steuerung von Lichtsignalanlagen in Straßennetzwerken*, Dissertation, Technische Universität Dresden, 2007.

[230] Carlos Gershenson, *Design and Control of Self-organizing Systems*, Ph.D. thesis, Faculteit Wetenschappen, Center Leo Apostel for Interdisciplinary Studies, Vrije Universiteit Brussel, Brussels, Belgium, March 2007.

[231] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe, *Self-Organization:Applied Multi-Agent Systems*, chapter Selforganizing traffic lights: A realistic simulation, pp. 41 – 49, Springer UK, London, UK, 2007, chapter 3.

[232] Carlos Gershenson, "Self-organizing traffic lights," *Complex Systems*, vol. 16, no. 1, pp. 29–53, 2005.

[233] Sascha Zechner, "Konzipierung und Entwicklung einer Fluss- oder Druck-basierten Steuerung für Organic Traffic Control (OTC)," Bachelor's thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, May 2009.

[234] Ana L. Bazzan, "A distributed approach for coordination of traffic signal agents," *Autonomous Agents and Multi-Agent Systems*, vol. 10, no. 2, pp. 131–164, 2005.

[235] Yaser Chaaban, Jörg Hähner, and Christian Müller-Schloer, "Towards robust hybrid central/self-organizing multi-agent systems," in *ICAART 2010 - Proceedings of the International Conference on Agents and Artificial Intelligence, Volume 2 - Agents, Valencia, Spain, January 22-24, 2010*, Joaquim Filipe, Ana L. N. Fred, and Bernadette Sharp, Eds. 2010, pp. 341–346, INSTICC Press.

[236] Matteo Vasirani and Sascha Ossowski, *Exploring the Potential of Multiagent Learning for Autonomous Intersection Control*, chapter 13, pp. 280–290, Idea Group Publishing, 2009.

[237] Ana L. C. Bazzan and Franziska Klügl, Eds., *Multi-Agent Systems for Traffic and Transportation Engineering*, Idea Group Publishing, Hershey, PA, US, 2009.

[238] As'ad Salkham, Raymond Cunningham, Anurag Garg, and Vinny Cahill, "A collaborative reinforcement learning approach to urban traffic control optimization," in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Sydney, NSW, Australia, December 9-12, 2008*, 2008, pp. 560–566.

[239] Raymond Cunningham, Anthony Harrington, and Vinny Cahill, "Middleware for next generation urban traffic control," in *Proceedings of the European Transport Conference (ETC)*, Oct. 2004.

[240] Jaime Barcelo and Jordi Casas, "Dynamic network simulation with aimsun," in *Proceedings of the International Symposium on Transport Simulation, Yokohama, Japan*, Amsterdam, NL, 2002, pp. 1–25, Kluwer.

[241] Transportation Research Board, "Highway capacity manual," Tech. Rep., National Research Council, Washington D.C., US, 2000.

[242] Martin Fellendorf, "Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority," in *Proceedings of the 64th Institute of Transportation Engineers Annual Meeting*, Dallas, US, 1994.

[243] Jens Ludmann, "Beeinflussung des Verkehrsablaufs auf Strasse – Analyse mit dem fahrzeugorientierten Verkehrssimulationsprogramm PELOPS," *Schriftenreihe Automobiltechnik, Institut für Kraftfahrwesen, Aachen*, 1998.

[244] Haifeng Xiao, Ravi Ambadipudi, John Hourdakis, and Panos Michalopoulos, "Methodology for selecting microscopic simulators: Comparative evaluation of AIMSUN and VISSIM," Tech. Rep. CTS 05-05, Department of Civil Engineering, University of Minnesota, 2005.

[245] F. Webster, *Traffic Signal Settings - Technical Paper No 39*, Road Research Laboratory, London, UK, 1959.

[246] Joseph M. Sussmann, *Perspectives on Intelligent Transportation Systems (ITS)*, Springer Verlag, Berlin, Germany, 2005.

[247] Sven Tomforde, Holger Prothmann, Fabian Rochner, Jürgen Branke, Jörg Hähner, Christian Müller-Schloer, and Hartmut Schmeck, "Decentralised Progressive Signal Systems for Organic Traffic Control," in *Proceedings of the 2nd IEEE International Conference on Self-Adaption and Self-Organization (SASO'08)*, Sven Brueckner, Paul Robertson, and Umesh Bellur, Eds. 2008, pp. 413–422, IEEE.

[248] Holger Prothmann, Jurgen 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*, vol. 2, no. 3, pp. 203 – 225, 2009.

[249] Ernest J. H. Chang, "Echo algorithms: Depth parallel operations on general graphs," *IEEE Transactions on Software Engineering*, vol. 8, no. 4, pp. 391–401, 1982.

[250] Leslie Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[251] David L. Mills, "Network Time Protocol (Version 3) - Specification, Implementation and Analysis," Tech. Rep. 90-6-1, Electrical Engineering Deptartment, University of Delaware, 1990.

[252] Sven Tomforde, Holger Prothmann, Jürgen Branke, Jörg Hähner, Christian Müller-Schloer, and Hartmut Schmeck, "Possibilities and limitations of decentralised traffic

control systems," in *2010 IEEE World Congress on Computational Intelligence (IEEE WCCI 2010)*. 2010, pp. 3298–3306, IEEE.

[253] Andrew S. Tanenbaum and Maarten van Steen, *Distributed Systems: Principles and Paradigms*, Pearson Education, 2nd edition edition, 2006.

[254] Carl Volhard, "Vorhersage der Verkehrsentwicklung für autonome Lichtsignal-Anlagensteuerungen," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, Hannover, De, September 2009.

[255] Ben Thancanamootoo and Matthew G. H. Bell, "Automatic Detection of Traffic Incidents on a Signal-Controlled Road Network," Tech. Rep. H7UNDT RR076, University of Newcastle upon Tyne, Deptartment of Civil Engineering, June 1988.

[256] Emily Parkanyi and Chi Xie, "A complete review of incident detection algorithms and their deployment: What works and what doesn't," Tech. Rep. NETCR 37, New England Transp. Consortium, Storrs, CT, February 2005.

[257] Xiao-Yuan Wang, Kai-Wang Zhang, and Xin-Yue Yang, "Research of the road traffic incident characteristics," in *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, August 2005, vol. 5, pp. 2688 – 2693.

[258] Peter T. Martin, Joseph Perrin, Blake Hansen, Ryan Kump, and Dan Moore, "Incident Detection Algorithm Evaluation," Tech. Rep. MPC-01-122, University of Utah, Prepared for Utah Department of Transportation, March 2001.

[259] Florian Mazur and Sigurdur Hafstein, "Verkehrslage via Internet: autobahn.NRW," *Logistik Management*, vol. 6, no. 4, pp. 60 – 69, 2004.

[260] Harold J. Payne and H.C. Knobel, "Development and Testing of Incident Detection Algorithm," FHWA Report FHWA RD 76 21, vol. 3, Federal Highway Administration, US Department of Transportation, Washington DC, US, 1976.

[261] Harold J. Payne and Samuel Coakley Tignor, "Freeway Incident-Detection Algorithms based on Decision Trees With States," TRB Research Record 682, Transportation Research Board, Washington DC, US, 1978.

[262] Lukas Klejnowski, "Design and implementation of an algorithm for the distributed detection of disturbances in traffic networks," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, August 2008.

[263] Horst F. Wedde, Sebastian Lehnhoff, Bernhard van Bonn, Zoltan Bay, Stefan Becker, Stefan Böttcher, Carl Brunner, Andreas Büscher, Thomas Fürst, Anca M. Lazarescu,

Elisei Rotaru, Sebastian Senge, Bernd Steinbach, Ferkan Yilmaz, and Timm Zimmermann, "Highly Dynamic and Adaptive Traffic Congestion Avoidance in Real-Time Inspired by Honey Bee Behavior," in *Mobilität und Echtzeit – Fachtagung der GI-Fachgruppe Echtzeitsysteme*, Peter Holleczek and Birgit Vogel-Heuser, Eds., Berlin / Heidelberg, DE, 2007, pp. 21 – 31, Springer Verlag.

[264] Andrew S. Tanenbaum, *Computer Networks*, Pearson Education, 4th edition, 2002.

[265] Johannes Joachim Lyda, "Dezentrale adaptive Routingverfahren in selbstorganisierten Verkehrsnetzen am Beispiel von Organic Traffic Control," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, October 2010.

[266] Radia Perlman, *Interconnections: Bridges and Routers*, Professional Computing Series. Addison Wesley, 2nd edition, 1999.

[267] Peter Hart, Nils Nilsson, and Bertram Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100 – 107, July 1968.

[268] John M. McQuillan, Ira Richer, and Eric C. Rosen, "The new routing algorithm for the ARPANET," in *Innovations in Internetworking*, Craig Partridge, Ed., pp. 119 – 127. Artech House, Inc., Norwood, MA, USA, 1988.

[269] Jeanna Matthews, *Computer Networking: Internet Protocols in Action*, John Wiley & Sons, 1st edition, 2005.

[270] TSS – Transport Simulation Systems, "Aimsun 5.1 microsimulator user's manual," 2008.

[271] Vehicle Certification Agency / UK Department for Transport, "New car fuel consumption & emission figures," 2009.

[272] Markus Weinreich, "Analyse von Sonderereignissen in urbanen Verkehrsnetzen unter Nutzung von Organic Traffic Control," Bachelor's thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, August 2009.

[273] Matti Siekkinen, Vera Goebel, Thomas Plagemann, Karl-Andre Skevik, Mark Banfield, and Igor Brusic, "Beyond the Future Internet–Requirements of Autonomic Networking Architectures to Address Long Term Future Networking Challenges," *Future Trends of Distributed Computing Systems, IEEE International Workshop*, pp. 89–98, 2007.

[274] Karin Anna Hummel, Andrea Hess, and Harald Meyer, "Mobilität im future internet," *Informatik Spektrum*, vol. 33, no. 2, pp. 143 – 159, April 2010.

[275] Mark Handley, "Why the internet only just works," *BT Technology Journal*, vol. 24, no. 3, pp. 119–129, July 2006.

[276] Thomas Kunz, *Reliable Multicasting in MANETs*, Ph.D. thesis, Carleton University, 2003.

[277] Martin Hoffmann, Michael Wittke, Jörg Hähner, and Christian Müller-Schloer, "Spatial Partitioning in Self-organising Camera Systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 1 – 10, August 2008.

[278] Sven Tomforde, Martin Hoffmann, Yvonne Bernard, Lukas Klejnowski, and Jörg Hähner, "POWEA: A System for Automated Network Protocol Parameter Optimisation Using Evolutionary Algorithms," in *Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, Stefan Fischer, Erik Maehle, and Rüdiger Reischuk, Eds. 2009, pp. 3177–3192, Gesellschaft für Informatik e.V. (GI).

[279] Sven Tomforde, Björn Hurling, and Jörg Hähner, "Dynamic control of mobile ad-hoc networks - network protocol parameter adaptation using organic network control," in *Proceedings of the 7th International Conference on Informatics in Control, Automation, and Robotics (ICINCO'10)*, Joaquim Filipe, Juan Andrade Cetto, and Jean-Louis Ferrier, Eds., Setubal, PT, 2010, vol. 1, pp. 28–35, INSTICC.

[280] Sven Tomforde, Björn Hurling, and Jörg Hähner, "Distributed Network Protocol Parameter Adaptation in Mobile Ad-Hoc Networks," in *Informatics in Control, Automation and Robotics*, Juan Andrade Cetto, Jean-Louis Ferrier, and Joaquim Filipe, Eds., vol. 89 of *Lecture Notes in Electrical Engineering*, pp. 91–104. Springer, Berlin Heidelberg, 2011.

[281] Sven Tomforde, Ioannis Zgeras, Jörg Hähner, and Christian Müller-Schloer, "Adaptive control of wireless sensor networks," in *Proceedings of the 7th International Conference on Autonomic and Trusted Computing (ATC'10)*, 2010, pp. 77 – 91.

[282] Sven Tomforde, Marcel Steffen, Jörg Hähner, and Christian Müller-Schloer, "Towards an Organic Network Control System," in *Proceedings of the 6th International Conference on Autonomic and Trusted Computing (ATC'09)*, Juan Gonzalez Nieto, Wolfgang Reif, Guojun Wang, and Jadwiga Indulska, Eds. 2009, pp. 2 – 16, Springer Verlag, Berlin, Heidelberg.

[283] Björn Hurling, Sven Tomforde, and Jörg Hähner, "Organic network control," in *Organic Computing - A Paradigm Shift for Complex Systems*, Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, Eds., chapter Chapter 6.1.11, pp. 611–612. Birkhäuser Verlag, 2011.

[284] Elias Weingärtner, Hendrik vom Lehn, and Klaus Wehrle, "A performance comparison of recent network simulators," in *ICC 2009: IEEE International Conference on Communications*, 2009.

[285] David Montana and Jason Redi, "Optimizing Parameters of a Mobile Ad-hoc Network Protocol with a Genetic Algorithm," in *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO'05)*, New York, NY, USA, 2005, pp. 1993–1998, ACM.

[286] Ethem M. Sözer, Milica Stojanovic, and John G. Proakis, "Initialization and routing optimization for ad-hoc underwater acoustic networks," in *Proceedings of Opnetwork'00*, 2000.

[287] Damla Turgut, Sajal Daz, Ramez Elmasri, and Begurnhan Turgut, "Optimizing Clustering Algorithm in Mobile Ad hoc Networks Using Genetic Algorithmic Approach," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02)*, 2002, pp. 62 – 66.

[288] Tijs van Dam and Koen Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, I. Akyildiz, D. Estrin, D. Culler, and M. Srivastava, Eds., New York, US, 2003, pp. 171–180, ACM.

[289] Kuo-Chun Huang, Xiangpeng Jing, and Dipankar Raychaudhuri, "Mac protocol adaptation in cognitive radio networks: An experimental study," *Computer Communications and Networks, International Conference on*, vol. 0, pp. 1–6, 2009.

[290] Andras Farago, Andrew D. Myers, Violet R. Syrotiuk, and Gergely V. Zaruba, "A new approach to MAC protocol optimization," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '00)*, 2000, vol. 3, pp. 1742–1746 vol.3.

[291] Andras Farago, Andrew D. Myers, Violet R. Syrotiuk, and Gergely V. Zaruba, "Meta-MAC protocols: automatic combination of MAC protocols to optimize performance for unknown conditions," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 9, pp. 1670–1681, Sep 2000.

[292] Pawan Goyal, Harrick M. Vin, Chia Shen, and Prashant J. Shenoy, "A reliable, adaptive network protocol for video transport," in *Proceedings of IEEE INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation*, 1996, vol. 3, pp. 1080–1090.

[293] Shimon Whiteson and Peter Stone, "Towards autonomic computing: adaptive network routing and scheduling," in *Proceedings of the International Conference on Autonomic Computing (ICAC'04)*, May 2004, pp. 286–287.

[294] Justin A. Boyan and Michael L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in Neural Information Processing Systems 6*, San Francisco, CA, US, 1994, pp. 671–678, Morgan Kaufmann.

[295] Jim Martin, Arne Nilsson, and Injong Rhee, "Delay-based congestion avoidance for TCP," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 356 – 369, 2003.

[296] Leonard Kleinrock and Fouad A.Tobagi, "Packet Switching in Radio Channels: Part I–Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1400–1416, dec 1975.

[297] Liliana Rosa, Antonia Lopes, and Luis Rodrigues, "Appia to R-Appia: Refactoring a Protocol Composition Framework for Dynamic Reconfiguration," Tech. Rep. 1, University of Lisbon, Department of Informatics, 1997.

[298] Hugo Miranda, Alexandre Pinto, and Luis Rodrigues, "Appia: A Flexible Protocol Kernel Supporting Multiple Coordinated Channels," in *Proceedings of the The 21st International Conference on Distributed Computing Systems (ICDCS '01)*, Washington, DC, USA, 2001, pp. 707 – 710, IEEE Computer Society.

[299] Matti A. Hiltunen, Richard D. Schlichting, Carlos A. Ugarte, and Gary T. Wong, "Survivability through customization and adaptability: the Cactus approach," in *Proceedings of the DARPA Information Survivability Conference and Exposition, 2000 (DISCEX '00)*, 2000, vol. 1, pp. 294 – 307.

[300] Shivakant Mishra, *Consul: a communication substrate for fault-tolerant distributed programs*, Ph.D. thesis, Tucson, AZ, USA, 1992.

[301] Robbert van Renesse, Ken Birman, Mark Hayden, Alexey Vaysburd, and David Karr, "Building Adaptive Systems Using Ensemble," *Software-Practice and Experience – Special issue on multiprocessor operating systems*, vol. 28, no. 9, pp. 963 – 979, 1998.

[302] Robbert van Renesse, Kenneth P. Birman, and Silvano Maffeis, "Horus: a flexible group communication system," *Communications of the ACM*, vol. 39, no. 4, pp. 76 – 83, 1996.

[303] Sergio Mena, André Schiper, and Pawel Wojciechowski, "A step towards a new generation of group communication systems," in *Middleware '03: Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, New York, NY, USA, 2003, pp. 414–432, Springer-Verlag New York, Inc.

[304] Thorsten Schöler and Christian Müller-Schloer, "First steps towards organic computing systems: monitoring an adaptive protocol stack with a fuzzy classifier system," in *Proceedings of the 2nd conference on Computing frontiers (CF'05)*, New York, NY, USA, 2005, pp. 10–20, ACM.

[305] Thorsten Schöler and Christian Müller-Schloer, "An observer/controller architecture for adaptive reconfigurable stacks," in *Proceedings of the 18th International Conference on Architecture of Computing Systems (ARCS'05)*, Berlin / Heidelberg, DE, 2005, pp. 139–153, Springer Verlag.

[306] Pradeep Sudame and Badri R. Badrinath, "On providing support for protocol adaptation in mobile wireless networks," *Mobile Networks and Applications*, vol. 6, no. 1, pp. 43 – 55, 2001.

[307] Tao Ye and Shivkumar Kalyanaraman, "An adaptive random search algorithm for optimizing network protocol parameters," Tech. Rep., Rensselaer Polytechnic Inst., 2001.

[308] Tao Ye and Shivkumar Kalyanaraman, "A recursive random search algorithm for network parameter optimization," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 3, pp. 44–53, 2004.

[309] Tao Ye, Hema T. Kaur, Shivkumar Kalyanaraman, and Murat Yuksel, "Large-scale network parameter configuration using an on-line simulation framework," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 777 – 790, 2008.

[310] Nikos Georganopoulos and Tim Lewis, "A framework for dynamic link and network layer protocol optimisation," *Mobile and Wireless Communications Summit, 2007. 16th IST*, pp. 1–5, 2007.

[311] Nikos Georganopoulos and Hamid Aghvami, "Performance Evaluation of Transport Protocols with Local Mobility Management," in *Personal Wireless Communications, IFIP-TC6 8th International Conference, PWC 2003, Venice, Italy, September 23-25, 2003, Proceedings*, Marco Conti, Silvia Giordano, Enrico Gregori, and Stephan Olariu, Eds., Berlin / Heidelberg, DE, 2003, vol. 2775 of *Lecture Notes in Computer Science*, pp. 251–260, Springer Verlag.

[312] Konstantinos Boukis, Nikos Georganopoulos, and Hamid Aghvami, "The Reconfigurable IP Mobility Component: Evaluation of Single Protocol Operation," in *Proceedings of the 64th IEEE Vehicular Technology Conference (VTC-2006 Fall)*, 25-28 2006, pp. 1 –5.

[313] William Su Sung-Ju, Sung ju Lee, and Mario Gerla, "Mobility prediction in wireless networks," in *Procceedings of the IEEE MILCOM*, 2000, pp. 491–495.

[314] Sungjoon Ahn and A. Udaya Shankar, "Adapting to route-demand and mobility (arm) in ad hoc network routing," in *9th International Conference on Network Protocols.* 2001, pp. 56–66, IEEE.

[315] Jeffrey Lowell Boleng, *Exploiting location information and enabling adaptive mobile ad hoc network protocols*, Ph.D. thesis, Golden, CO, USA, 2002.

[316] Young-Bae Ko and Nitin H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *Wireless Networking*, vol. 6, no. 4, pp. 307 – 321, 2000.

[317] Oliver Stanze, Martina Zitterbart, and Christian Koch, "Mobility Adaptive Self-Parameterization of Routing Protocols for Mobile Ad Hoc Networks," in *Proceedings of IEEE Wireless Communication and Networking Conference (WCNC)*, Las Vegas, USA, Apr. 2006, vol. 1, pp. 276–281.

[318] Hubert Zimmermann, "Osi reference model — the iso model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, April 1980.

[319] Thomas Kunz, "Multicasting in mobile ad-hoc networks: achieving high packet delivery ratios," in *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative Research (CASCON'03)*, 2003, pp. 156–170.

[320] Kaveh Pahlavan and Prashant Krishnamurthy, *Principles of Wireless Networks: A Unified Approach*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[321] Gregory F. Lawler and Vlada Limic, *Random walk : a modern introduction*, Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2010.

[322] Cauligi S. Raghavendra, Taieb Znati, and Krishna M. Sivalingam, Eds., *Wireless Sensor Networks*, ERCOFTAC Series. Springer Netherlands, 2nd edition, 2004.

[323] Ioannis Zgeras, "Entwurf und Implementierung einer Agenten-basierten Simulationsumgebung für adaptive und robuste Sensornetzprotokolle," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, October 2009.

[324] Hock Beng Lim, Vinh The Lam, Mao Ching Foo, and Yulian Zeng, "Adaptive Distributed Resource Allocation in Wireless Sensor Networks," White paper (technical report), National University of Singapore, 2005.

[325] Hock-Beng Lim, Vinh The Lam, Mao Ching Foo, and Yulian Zeng, "An Adaptive Distributed Resource Allocation Scheme for Sensor Networks," in *Proceedings of the International Conference on Mobile ad-hoc and sensor networks (MSN'06)*, 2006, pp. 770 – 781.

[326] Johan A. Pouwelse, Pawel Garbacki, Dick H. J. Epema, and Henk J. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Peer-to-Peer Systems IV, 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25,*

2005, Revised Selected Papers, Miguel Castro and Robbert van Renesse, Eds., Berlin / Heidelberg, 2005, vol. 3640 of *Lecture Notes in Computer Science*, pp. 205–216, Springer Verlag.

[327] Ipoque, "Internet study report 2008/2009," http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009, 2009.

[328] Ralf Steinmetz and Klaus Wehrle, Eds., *Peer-to-Peer Systems and Applications*, Lecture Notes in Computer Science. Springer Verlag, Berlin / Heidelberg, DE, 1st edition, 2005.

[329] Web, "The vuze bittorrent client," http://www.vuze.com/, Vuze, Inc., 2010.

[330] Kolja Eger, "Simulation of BitTorrent Peer-to-Peer Networks in ns-2," http://www.tu-harburg.de/et6/research/bittorrentsim/index.html.

[331] Christoph König, "Analyse und Umsetzung von kollaborativem Lernen in einem verteilten Netzwerksteuerungssystem," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, August 2009.

[332] Sven Tomforde, "A Case Study for an Organic Production System," Technical Report SRA-01-2011, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture Group, Hannover, DE, July 2011.

[333] IBM Corporation, "System Z webpage," Online, `http://www-03.ibm.com/systems/de/z/`, 2010.

[334] Miao Li, "Automatisiertes Data Mining von kontinuierlichen gesammelten System-Kennzahlen im IBM System Z Umfeld," Master thesis, Leibniz Universität Hannover, Institute for Systems Engineering, Sytem and Computer Architecture Group (SRA), Hannover, DE, May 2009.

[335] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo, "Discovery of Frequent Episodes in Event Sequences," *Data Mining and Knowledge Discovery, Springer Netherlands*, vol. 1, no. 3, pp. 259–289, September 1997.

[336] Sven Tomforde, Andreas Brameshuber, Jörg Hähner, and Christian Müller-Schloer, "Restricted On-line Learning in Real-world Systems," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2011, New Orleans, LA, USA, 5-8 June, 2011*, New Orleans, LA, US, 2011, pp. 1628 – 1635, IEEE.

[337] Andreas Brameshuber, "Untersuchung 2-schichtigen Lernens auf mathematisch modellierten Fitnesslandschaften," Bachelor's thesis, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, March 2011.

[338] Emre Cakar, Sven Tomforde, and Christian Müller-Schloer, "A role-based imitation algorithm for the optimisation in dynamic fitness landscapes," in *IEEE Swarm Intelligence Symposium, 2011. SIS 2011*, Paris, France, 2011, pp. 139–146.

[339] Rahmi Akçelik, "Traffic signals: Capacity and timing analysis," Tech. Rep. 123, Australian Road Research Board, 1981.

[340] Björn Hurling, Sven Tomforde, and Jörg Hähner, "A generic architecture for restricted on-line learning of network protocol parameters," in *Proceedings of the first Workshop on Self-Adaptive Networking (SAN'10), held in conjunction with 4th IEEE International Conference on Self-Adaptive and Self-Organising Systems (SASO'10)*, 2010.

[341] Stefan Rudolph, "Konvergenz in Learning Classifier Systems," Seminar paper, Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture, July 2010.

[342] Emre Cakar, Nugroho Fredivianus, Jörg Hähner, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck, "Aspects of Learning in OC Systems," in *Organic Computing - A Paradigm Shift for Complex Systems*, Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, Eds., incollection 3.1, pp. 237 – 251. Birkhäuser, June 2011.

Appendix

Appendix A: Classification of Machine Learning

Techniques

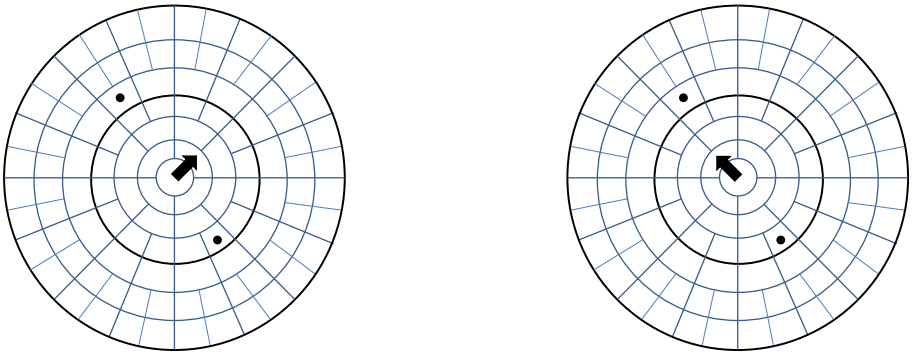
Part A of the appendix lists the result of the classification process of machine learning techniques in the context of Chapter 4.1. Therefore, the general characteristics as introduced in Chapter 4.1.2 serve as basis for the classification. The following list repeats the particular characteristics and assign an ID to each of them. This ID is used in the following table. Within this table, the scale as introduced for Chapter 2 is applied by using five classes: "++", "+", "0", "−", and "−−". The first one denotes a full match of the requirement by the particular technique, "0" states that the requirement might be satisfiable under certain circumstances, and "−−" states that the requirement is not fulfilled.

- **A** Learning at runtime
- **B** Pre-training
- **C** Online sandboxing
- **D** Traceability by engineer
- **E** Noisy sensor data
- **F** Fast reaction time
- **G** Large configuration space
- **H** Large situation space
- **I** No final target state
- **J** Subsequent state is not deterministic
- **K** Non-deterministic (on-line) reward
- **L** Limited learning cycles

Technique	A	B	C	D	E	F	G	H	I	J	K	L
Decision Trees	−−	++	−−	+	0	++	−−	−	+	+	++	0
Bayesian Belief Networks	−−	++	−−	0	0	++	−−	−	+	+	++	0
Association rule learning	−−	++	−−	++	0	+	−−	−−	+	+	++	0
Inductive logic programming	−−	++	−−	++	0	+	−−	−−	+	+	++	0
Support vector machines	−−	++	−−	−	0	+	−	+	+	+	++	0
Artificial Neural Networks	−−	++	−−	−−	++	++	++	++	++	++	++	0
(Modified) ANN	−−	++	−−	−−	++	++	++	++	++	++	++	0
Learning Classifier Systems	++	++	−−	++	++	++	++	++	++	++	++	−
(Modified) LCS	++	++	++	++	++	++	++	++	++	++	++	++
Fuzzy Classifier Systems	++	++	−−	0	++	++	++	++	++	++	++	−
(Modified) FCS	++	++	++	0	++	++	++	++	++	++	++	++
Q-Learning	++	++	−−	−−	++	++	−−	−	+	+	+	−
TD-Learning	++	++	−−	−−	++	++	−−	−	+	+	+	−

Table 9.1: Classification of machine learning techniques

Appendix B: Comparison of Optimisation Techniques



(a) Situation 1 (b) Situation 2

Figure 9.1: Exemplary situations (Situation 1 and 2)

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	49	3	12.9429	0.8049	10.0367	0.8242	248.16	96.90
GA	83	61	13.1183	0.7004	11.6277	1.0250	78.10	40.25
HS	28	23	12.4767	1.1038	10.7654	0.8490	156.69	103.43
PSO	23	14	12.6921	0.6658	10.6435	0.8263	152.99	80.70
RD	0	1	10.6144	0.4651	9.8613	0.6608	160.67	84.05
SA	1	1	9.8204	1.3502	8.2905	1.5520	321.21	133.33

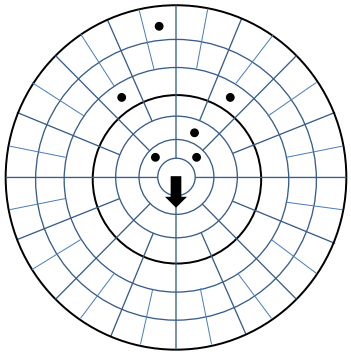
Table 9.2: Results for Situation 1

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	33	9	13.3011	0.7076	10.2740	0.8754	202.81	91.10
GA	18	76	13.2470	0.5791	11.9748	0.9573	67.99	54.73
HS	24	28	12.6763	1.0206	10.9437	0.9711	147.47	102.24
PSO	16	17	12.7442	0.6978	10.6999	0.9171	142.22	71.55
RD	0	4	11.1281	0.6796	10.1905	0.6022	276.74	137.37
SA	4	4	10.1105	1.6329	8.0619	1.6475	252.16	167.63

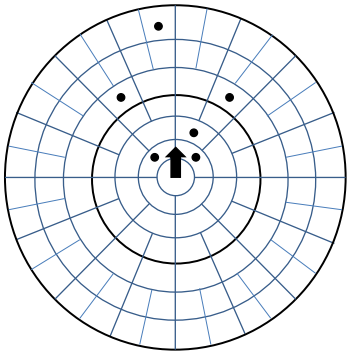
Table 9.3: Results for Situation 2

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	34	22	22.8668	1.9424	15.4908	2.6483	135.48	66.68
GA	85	91	24.9561	1.4507	20.1839	2.3254	48.30	27.14
HS	29	60	22.4558	2.2356	17.8443	2.3607	82.62	60.55
PSO	44	48	23.6499	1.8405	13.3274	2.8862	90.53	52.55
RD	3	22	19.3371	1.9788	16.0749	2.1606	173.44	116.82
SA	5	7	15.8673	4.7608	8.5769	4.7380	267.23	158.68

Table 9.4: Results for Situation 3



(a) Situation 3

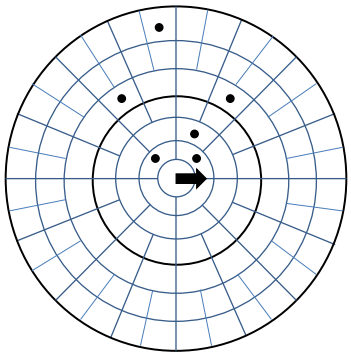


(b) Situation 4

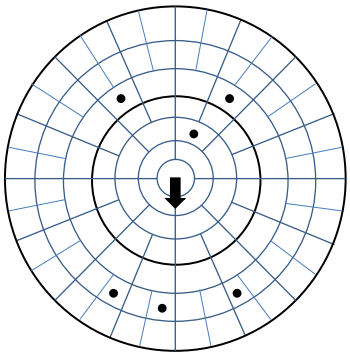
Figure 9.2: Exemplary situations (Situation 3 and 4)

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	35	24	21.2105	1.6065	13.6879	2.4773	124.45	52.13
GA	61	96	22.9522	2.2211	19.6722	2.3403	42.10	20.76
HS	52	61	21.9391	2.7724	16.0406	3.2818	92.54	80.22
PSO	51	39	22.0187	1.6402	14.5536	2.9352	108.91	60.94
RD	8	9	15.8486	2.5044	12.5327	2.5351	204.94	136.85
SA	8	10	14.6737	5.5508	8.2697	4.1664	246.48	154.08

Table 9.5: Results for Situation 4



(a) Situation 5



(b) Situation 6

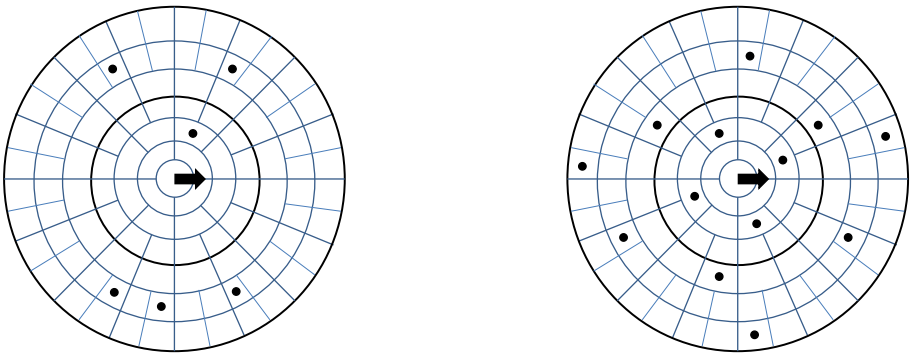
Figure 9.3: Exemplary situations (Situation 5 and 6)

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	42	21	23.0487	1.0968	14.6087	3.2400	134.76	74.47
GA	86	89	24.3678	1.1340	21.0835	2.7751	47.98	28.90
HS	48	58	22.9355	1.8952	18.0346	3.4676	83.20	57.16
PSO	58	45	23.6461	1.1240	16.2594	3.4386	105.00	58.43
RD	3	21	18.8036	2.2663	14.7831	2.9576	185.37	126.92
SA	5	9	15.2561	6.0656	8.8607	4.7522	244.79	159.56

Table 9.6: Results for Situation 5

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	4	35	28.1198	1.0925	21.1208	3.7389	119.92	68.42
GA	43	84	30.7414	2.4902	26.5917	3.5866	46.42	34.92
HS	2	65	28.1439	1.7005	24.3454	2.6449	80.02	82.07
PSO	34	51	30.5453	2.5959	22.9540	3.6205	94.76	55.48
RD	0	24	25.5172	1.4986	20.4808	3.4786	186.38	133.19
SA	2	11	23.1877	5.6783	14.7351	5.6662	236.61	150.69

Table 9.7: Results for Situation 6



(a) Situation 7 (b) Situation 8

Figure 9.4: Exemplary situations (Situation 7 and 8)

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	3	9	27.4569	1.0056	21.9374	1.7774	183.30	74.83
GA	47	85	29.5304	2.2898	26.9274	2.2272	56.39	38.94
HS	5	39	27.0549	1.4978	23.5525	2.1253	133.67	103.77
PSO	21	24	28.7896	1.8344	23.0679	1.8054	124.23	60.65
RD	0	5	23.4211	1.3423	21.2852	1.5042	240.28	155.71
SA	2	5	24.0763	2.6695	16.5265	5.1809	305.77	149.35

Table 9.8: Results for Situation 7

	SR(500)	SR(83)	AP (500)	Std. Dev.	AP(83)	Std. Dev.	NC	Std. Dev.
DE	22	13	23.2290	7.2118	11.8427	4.1119	195.01	94.96
GA	74	67	27.2051	5.4273	16.9912	5.7171	72.62	45.59
HS	59	48	26.3289	5.7101	15.1670	6.3728	104.96	71.84
PSO	32	18	24.4271	6.6363	12.5649	4.2316	154.13	82.54
RD	15	10	17.5451	8.1467	12.1552	4.7833	196.67	124.00
SA	19	6	17.9791	10.7517	8.0483	4.1706	285.69	142.12

Table 9.9: Results for Situation 8

Appendix C: Organic Traffic Control

C.1: Scenario located at Hamburg, Germany

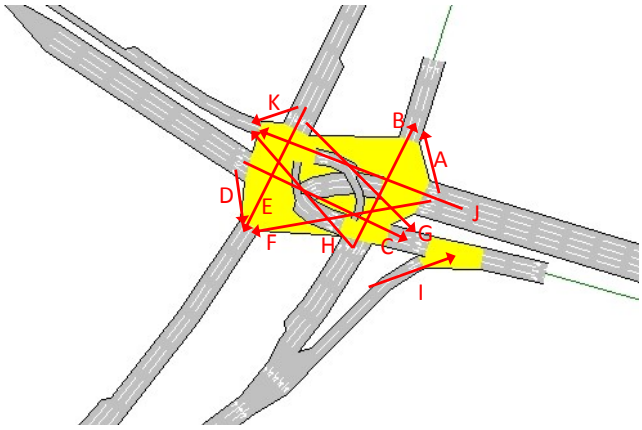


Figure 9.5: Turnings of the investigated intersection

Signal Group ID	S1	S2	S3	S4	S6	S7	S8	S10
Contains turning	A+J	H	F	C+D	G	I	B	E+G+K

Table 9.10: Relation between signal groups and turning movements for node K3

Phase ID	Contained signal groups	Phase duration
P1	S3 + S7	1 *s*
P2	S7	5 *s*
P3	S7 + S8	3 *s*
P4	S7 + S8 + S10	34 *s*
P5	S7 + S8	3 *s*
P6	–	3 *s*
P7	S2	2 *s*
P8	S2 + S4 + S6	5 *s*
P9	S1 + S2 + S4 + S6	9 *s*
P10	S1 + S2 + S6	4 *s*
P11	S1 + S2 + S3 + S6	3 *s*
P12	S1 + S2 + S3	7 *s*
P13	S1 + S2 + S3 + S7	1 *s*
P14	S2 + S3 + S7	6 *s*
P15	S3 + S7	4 *s*

Table 9.11: List of signal groups, their assignment to phases, and the green time durations for node K3

	Borsteler Chaussee	Alsterkrugchaussee	Deelböge	Rosenbrook
Borsteler Chaussee			8,754	2,016
Alsterkrugchaussee	1,080		1,738	14,209
Deelböge	7,7,38	690		13,987
Rosenbrook	1,618	13,307	13,586	
Total 78,723	10,436	13,997	24,078	30,212

Table 9.12: Aggregated traffic demand for node K3 (number of vehicles)

C.2: Stadium scenario located at Hannover, Germany

The following tables list the traffic demands as considered for the simulation of the stadium event in Chapter 6.5.2. The data has been provided by the Landeshauptstadt Hannover, Fachbereich Tiefbau (Bereich Koordinierung und Verkehr) and the Verkehrsmanagementzentrale Niedersachsen (Region Hannover). The traffic demands are derived from a census performed at the 9th of May, 2009. Additionally, turning probabilities for in-between intersections (and approaches to car parks) where no actual data exist, have been approximated [272]. In some tables, the roads of an intersection appear twice – each with an additional attribute. Thereby, E (*east*), W (*west*), N (*north*), and S (*south*) specify the part of the road if it crosses the intersection.

Traffic demands for intersection Aegidientorplatz / Friedrichswall / Willy-Brandt-Allee / Osterstr.

	Aegidientorplatz	Willy-Brandt-Allee	Friedrichswall	Osterstr.
Aegidientorplatz		253	1162	230
Willy-Brandt-Allee	176		38	145
Friedrichswall	740	43		175
Osterstr.	199	57	132	
Total	**1115**	**353**	**1332**	**550**

Table 9.13: Arrival 12 – 13 o'clock

	Aegidientorplatz	Willy-Brandt-Allee	Friedrichswall	Osterstr.
Aegidientorplatz		268	1271	237
Willy-Brandt-Allee	100		44	97
Friedrichswall	1008	68		143
Osterstr.	235	56	130	
Total	**1343**	**392**	**1445**	**447**

Table 9.14: Arrival 13 – 14 o'clock

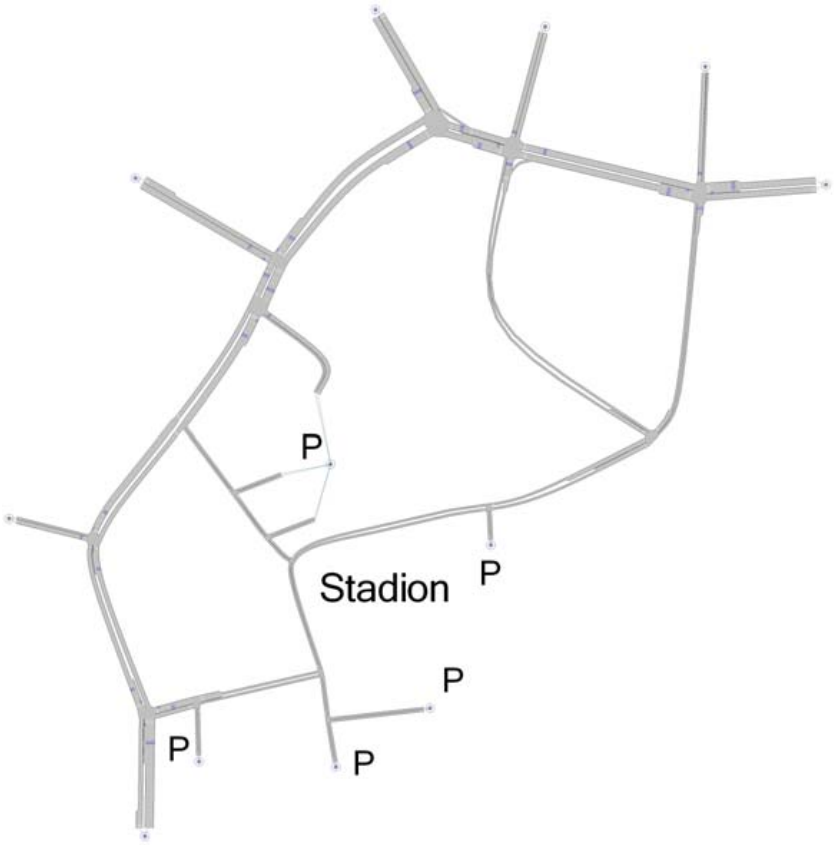


Figure 9.6: The investigated road-network of the stadium area at Hannover, Germany

	Aegidientorplatz	Willy-Brandt-Allee	Friedrichswall	Osterstr.
Aegidientorplatz		253	1431	199
Willy-Brandt-Allee	128		76	118
Friedrichswall	991	127		141
Osterstr.	247	61	141	
Total	**1366**	**441**	**1648**	**458**

Table 9.15: Arrival 14 – 15 o'clock

	Aegidientorplatz	Willy-Brandt-Allee	Friedrichswall	Osterstr.
Aegidientorplatz		233	942	102
Willy-Brandt-Allee	210		97	123
Friedrichswall	1277	56		200
Osterstr.	250	49	113	
Total	**1737**	**338**	**1152**	**425**

Table 9.16: Departure 17:30 – 18:30 o'clock

Traffic demands for intersection Friedrichswall / Culemannstr. / Karmarschstr.

	Aegidientorplatz	Willy-Brandt-Allee	Friedrichswall	Osterstr.
Aegidientorplatz		291	919	99
Willy-Brandt-Allee	300		68	139
Friedrichswall	876	45		135
Osterstr.	205	43	101	
Total	**1381**	**379**	**1088**	**373**

Table 9.17: Departure 18:30 – 19:30 o'clock

	Friedrichswall (E)	Culemannstr.	Friedrichswall (W)	Karmarschstr.
Friedrichswall (E)			320	35
Culemannstr.	81		240	35
Friedrichswall (W)	768	229		822
Karmarschstr.	60	144		
Total	**909**	**373**	**560**	**892**

Table 9.18: Arrival 12 – 13 o'clock

	Friedrichswall (E)	Culemannstr.	Friedrichswall (W)	Karmarschstr.
Friedrichswall (E)			356	39
Culemannstr.	78		227	34
Friedrichswall (W)	1046	360		806
Karmarschstr.	72	99		
Total	**1196**	**459**	**583**	**879**

Table 9.19: Arrival 13 – 14 o'clock

	Friedrichswall (E)	Culemannstr.	Friedrichswall (W)	Karmarschstr.
Friedrichswall (E)			455	51
Culemannstr.	62		398	27
Friedrichswall (W)	1016	408		680
Karmarschstr.	84	94		
Total	**1162**	**502**	**853**	**758**

Table 9.20: Arrival 14 – 15 o'clock

	Friedrichswall (E)	Culemannstr.	Friedrichswall (W)	Karmarschstr.
Friedrichswall (E)			264	29
Culemannstr.	86		401	10
Friedrichswall (W)	1084	267		651
Karmarschstr.	87	144		
Total	**1257**	**411**	**665**	**690**

Table 9.21: Departure 17:30 – 18:30 o'clock

Traffic demands for intersection Friedrichswall / Leibnizufer / Lavesallee

	Friedrichswall (E)	Culemannstr.	Friedrichswall (W)	Karmarschstr.
Friedrichswall (E)			315	35
Culemannstr.	102		342	11
Friedrichswall (W)	860	236		555
Karmarschstr.	58	124		
Total	**1020**	**360**	**657**	**601**

Table 9.22: Departure 18:30 – 19:30 o'clock

	Leibnizufer	Friedrichswall	Lavesallee
Leibnizufer		679	289
Friedrichswall	284		664
Lavesallee	310	659	
Total	**594**	**1338**	**953**

Table 9.23: Arrival 12 – 13 o'clock

	Leibnizufer	Friedrichswall	Lavesallee
Leibnizufer		823	308
Friedrichswall	318		743
Lavesallee	375	930	
Total	**693**	**1753**	**1051**

Table 9.24: Arrival 13 – 14 o'clock

	Leibnizufer	Friedrichswall	Lavesallee
Leibnizufer		808	388
Friedrichswall	500		1166
Lavesallee	351	896	
Total	**851**	**1704**	**1554**

Table 9.25: Arrival 14 – 15 o'clock

	Leibnizufer	Friedrichswall	Lavesallee
Leibnizufer		546	231
Friedrichswall	332		775
Lavesallee	739	1130	
Total	**1071**	**1676**	**1006**

Table 9.26: Departure 17:30 – 18:30 o'clock

	Leibnizufer	Friedrichswall	Lavesallee
Leibnizufer		637	235
Friedrichswall	331		774
Lavesallee	425	673	
Total	**756**	**1310**	**1009**

Table 9.27: Departure 18:30 – 19:30 o'clock

Traffic demands for intersection Lavesallee / Gustav-Bratke-Allee

	Lavesallee (N)	Friedrichswall	Lavesallee (S)
Lavesallee (N)		253	707
Friedrichswall	229		96
Lavesallee (S)	641	113	
Total	**870**	**366**	**803**

Table 9.28: Arrival 12 – 13 o'clock

	Lavesallee (N)	Friedrichswall	Lavesallee (S)
Lavesallee (N)		267	791
Friedrichswall	211		111
Lavesallee (S)	941	145	
Total	**1152**	**412**	**902**

Table 9.29: Arrival 13 – 14 o'clock

	Lavesallee (N)	Friedrichswall	Lavesallee (S)
Lavesallee (N)		323	1261
Friedrichswall	219		135
Lavesallee (S)	927	148	
Total	**1146**	**471**	**1396**

Table 9.30: Arrival 14 – 15 o'clock

	Lavesallee (N)	Friedrichswall	Lavesallee (S)
Lavesallee (N)		295	830
Friedrichswall	156		91
Lavesallee (S)	1692	228	
Total	**1848**	**523**	**921**

Table 9.31: Departure 17:30 – 18:30 o'clock

	Lavesallee (N)	Friedrichswall	Lavesallee (S)
Lavesallee (N)		289	811
Friedrichswall	169		97
Lavesallee (S)	646	114	
Total	**815**	**403**	**908**

Table 9.32: Departure 18:30 – 19:30 o'clock

Traffic demands for intersection Lavesallee / Am Waterlooplatz

	Lavesallee (N)	Am Waterlooplatz	Lavesallee (S)
Lavesallee (N)		120	683
Am Waterlooplatz	46		146
Lavesallee (S)	708	124	
Total	**754**	**244**	**829**

Table 9.33: Arrival 12 – 13 o'clock

	Lavesallee (N)	Am Waterlooplatz	Lavesallee (S)
Lavesallee (N)		135	767
Am Waterlooplatz	60		168
Lavesallee (S)	1026	184	
Total	**1086**	**319**	**935**

Table 9.34: Arrival 13 – 14 o'clock

	Lavesallee (N)	Am Waterlooplatz	Lavesallee (S)
Lavesallee (N)		194	1102
Am Waterlooplatz	99		316
Lavesallee (S)	976	235	
Total	**1075**	**429**	**1418**

Table 9.35: Arrival 14 – 15 o'clock

	Lavesallee (N)	Am Waterlooplatz	Lavesallee (S)
Lavesallee (N)		138	783
Am Waterlooplatz	301		171
Lavesallee (S)	1619	336	
Total	**1920**	**474**	**954**

Table 9.36: Departure 17:30 – 18:30 o'clock

	Lavesallee (N)	Am Waterlooplatz	Lavesallee (S)
Lavesallee (N)		136	767
Am Waterlooplatz	93		125
Lavesallee (S)	667	93	
Total	**760**	**229**	**892**

Table 9.37: Departure 18:30 – 19:30 o'clock

Traffic demands for intersection Ritter-Brüning-Str. / Allerweg

	Ritter-Brüning-Str. (N)	Allerweg	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		125	705
Allerweg	100		100
Ritter-Brüning-Str. (S)	832	147	
Total	**932**	**272**	**805**

Table 9.38: Arrival 12 – 13 o'clock

	Ritter-Brüning-Str. (N)	Allerweg	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		125	710
Allerweg	100		100
Ritter-Brüning-Str. (S)	1210	214	
Total	**1310**	**339**	**810**

Table 9.39: Arrival 13 – 14 o'clock

	Ritter-Brüning-Str. (N)	Allerweg	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		213	1205
Allerweg	100		100
Ritter-Brüning-Str. (S)	1211	214	
Total	**1311**	**427**	**1305**

Table 9.40: Arrival 14 – 15 o'clock

	Ritter-Brüning-Str. (N)	Allerweg	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		143	811
Allerweg	80		80
Ritter-Brüning-Str. (S)	1955	345	
Total	**2035**	**488**	**891**

Table 9.41: Departure 17:30 – 18:30 o'clock

	Ritter-Brüning-Str. (N)	Allerweg	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		134	759
Allerweg	80		80
Ritter-Brüning-Str. (S)	760	134	
Total	**840**	**268**	**839**

Table 9.42: Departure 18:30 – 19:30 o'clock

Traffic demands for intersection Ritter-Brüning-Str. / Stadionbrücke

	Ritter-Brüning-Str. (N)	Stadionbrücke	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		211	594
Stadionbrücke	334		215
Ritter-Brüning-Str. (S)	645	161	
Total	**979**	**372**	**809**

Table 9.43: Arrival 12 – 13 o'clock

	Ritter-Brüning-Str. (N)	Stadionbrücke	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		262	709
Stadionbrücke	306		140
Ritter-Brüning-Str. (S)	1118	280	
Total	**1424**	**542**	**849**

Table 9.44: Arrival 13 – 14 o'clock

	Ritter-Brüning-Str. (N)	Stadionbrücke	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		251	790
Stadionbrücke	258		94
Ritter-Brüning-Str. (S)	1167	292	
Total	**1425**	**543**	**884**

Table 9.45: Arrival 14 – 15 o'clock

	Ritter-Brüning-Str. (N)	Stadionbrücke	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		137	409
Stadionbrücke	62		174
Ritter-Brüning-Str. (S)	2300	256	
Total	**2362**	**393**	**583**

Table 9.46: Departure 17:30 – 18:30 o'clock

	Ritter-Brüning-Str. (N)	Stadionbrücke	Ritter-Brüning-Str. (S)
Ritter-Brüning-Str. (N)		90	318
Stadionbrücke	100		184
Ritter-Brüning-Str. (S)	894	99	
Total	**994**	**189**	**492**

Table 9.47: Departure 18:30 – 19:30 o'clock

Remaining intersection without traffic demands

	Beuermannstr. (N)	Schützenplatz	Beuermannstr. (S)
Beuermannstr. (N)		50 %	50 %
Schützenplatz	50 %		50 %
Beuermannstr. (S)	50 %	50 %	

Table 9.48: Approach to car park Schützenplatz from Beuermannstr.

	Beuermannstr. (N)	Arthur-Menge-Ufer	Beuermannstr. (S)
Beuermannstr. (N)		50 %	50 %
Arthur-Menge-Ufer	50 %		50 %
Beuermannstr. (S)	50 %	50 %	

Table 9.49: Intersection Beuermannstr. / Arthur-Menge-Ufer

	Arthur-Menge-Ufer (W)	Seuferallee	Arthur-Menge-Ufer (E)
Arthur-Menge-Ufer (W)		50 %	50 %
Seuferallee	50 %		50 %
Arthur-Menge-Ufer (E)	50 %	50 %	

Table 9.50: Approach to car park Seuferallee from Arthur-Menge-Ufer

	Arthur-Menge-Ufer	Culemannstr.	Willy-Brandt-Allee
Arthur-Menge-Ufer		50 %	50 %
Culemannstr.	0 %		100 %
Willy-Brandt-Allee	0 %	100 %	

Table 9.51: Intersection Arthur-Menge-Ufer / Culemannstr. / Willy-Brandt-Allee

	Stadionbrücke (W)	Stammestr.	Stadionbrücke (E)
Stadionbrücke (W)		50 %	50 %
Stammestr.	50 %		50 %
Stadionbrücke (E)	50 %	50 %	

Table 9.52: Approach to car park Stammestr. from Stadionbrücke

	Stadionbrücke	F.-W.-Fricke Weg	Beuermannstr.
Stadionbrücke		50 %	50 %
F.-W.-Fricke Weg	50 %		50 %
Beuermannstr.	50 %	50 %	

Table 9.53: Intersection Stadionbrücke / F.-W.-Fricke Weg / Beuermannstr.

	F.-W.-Fricke Weg	Lodemannweg	F.-W.-Fricke Weg (P)
F.-W.-Fricke Weg		50 %	50 %
Lodemannweg	50 %		50 %
F.-W.-Fricke Weg (P)	100 %	0 %	

Table 9.54: Approach to car park Lodemannweg from F.-W.-Fricke Weg

	Ritter-Brüning-Str.	Beuermannstr.	Lavesallee
Ritter-Brüning-Str.		20 %	80 %
Beuermannstr.	0 %		100 %
Lavesallee	100 %	0 %	

Table 9.55: Intersection Ritter-Brüning-Str. / Beuermannstr. / Lavesallee (Arrival)

	Ritter-Brüning-Str.	Beuermannstr.	Lavesallee
Ritter-Brüning-Str.		5 %	95 %
Beuermannstr.	0 %		100 %
Lavesallee	100 %	0 %	

Table 9.56: Intersection Ritter-Brüning-Str. / Beuermannstr. / Lavesallee (Departure)

Sven Tomforde

Address	Bennostr. 16, 30451 Hanover, Germany
Telephone	+49 (0)511-43 74 540
Mobile	+49 (0)177-317 61 00
E-Mail	sven.tomforde@web.de
Date of birth	17. December 1978 in Buchholz i.d.N.
Nationality	German

PhD AND UNIVERSITY

06/2007 – 10/2011 **Dr.-Ing. degree at the Faculty for Electrical Engineering and Computer Science at the Leibniz University of Hanover, Germany**
PhD in Organic Computing at the Institute for Systems Engineering (ISE) in association with Prof. Müller-Schloer:
"An Architectural Framework for Self-configuration and Self-improvement at Runtime "
(Advisor on 17 Diploma/Master/Bachelor theses and coordinator for seven scientific aids)

04/2005 – 05/2007 **Degree in Computer Science** at the Leibniz University in Hannover, Germany (M. Sc.)

05/2005 – 12/2005 **M.Sc. at the Leibniz University in Hannover and the University of Bristol**
Thesis in association with Prof. Brehm and Dr. Muller:
"Author Disambiguation for Scientific Papers"

09/2006 – 05/2007 Term abroad (Erasmus) at the University of Bristol, United Kingdom**, Computer Science**

09/2002 – 09/2004 **Degree in Business Informatics** at the University of Applied Sciences and Arts Hannover, Germany (Diploma / FH)
Thesis in association with Prof. Walenda:
"Durchführung einer empirischen Studie und Entwicklung eines intranetbasierten Systems zur Verbesserung des Wissens-austauschs im Supply Chain Management der Continental AG"

08/1999 – 08/2002 **Degree in Business Economics** at the Berufsakademie für Bankwirtschaft, Hannover, Germany (Betriebswirt / BA)

Degree in Banking at the Industrie- und Handelskammer (Chamber of Industry and Commerce) Hannover, Germany (Bankkaufmann / Bank business management assistant)

1991 – 1998 **Grammar school** (Gymnasium Winsen), Winsen/Luhe; Abitur 1998 (A-levels)

EXPERIENCE

06/2007 – 10/2011 **Research project "Organic Traffic Control (OTC)"**
Development of a decentralised traffic control system
Technologies: *C++, JAVA, Python, Aimsun, MASON*

03/2004 – 09/2004	**Internship** at AWD AG in Hannover, Germany Department: Application Management Technologies: *JAVA, J2EE, MS Office, SQL*
11/2003 – 03/2004	**Research project "Kalymnos"** at Fachhochschule Hannover Development of a component-based application for a company from the financial sector Technologies: *JAVA, SQL*
03/2004 – 09/2004	**Internship** at Continental AG in Hannover, Germany Department: Supply Chain Management Technologies: *HTML, MS Office*
08/1999 – 07/2001	**Trainee at** Volksbank Syke eG, Syke, Germany Bank business management assistant
07/1998 – 07/1999	**Civilian Service at German Red Cross,** Winsen/Luhe, Germany Emergency medical technician

TEACHING EXPERIENCE

09/2007 – 10/2011	**In charge of the SRA Tutorial "Organic Computing"** Technologies: *JAVA, NetLOGO*
09/2007 – 10/2011	**In charge of the SRA Student Seminar "Organic Computing"**

ADDITIONAL EDUCATION

02/2009	**Participation** in the International School of **Design of Collective Intelligence** at the Lorentz Center, Leiden, Netherlands
11/2007	**Participation** in the International Winter School of **Self-Organisation in Embedded Systems** at Schloss Dagstuhl, Wadern, Germany
09/2003 – 03/2004	**Project** for the Volkswagen AG Wolfsburg, Germany **Development of an ASP-based Rights Management for Intranets**

LANGUAGES

German:	Native language
English:	Fluent both orally and in writing (thesis written in English)
Latin:	Latin certificate (school time)
French:	Basic knowledge