



Intelligent
Embedded Systems

18

S. Tomforde | C. Krupitzer (Eds.)

Organic Computing

Doctoral Dissertation Colloquium 2020



Organic Computing

kassel
university



press



Intelligent
Embedded Systems

Band 18

Herausgegeben von
Prof. Dr. Bernhard Sick, Universität Kassel

Organic Computing

Doctoral Dissertation Colloquium 2020

S. Tomforde, C. Krupitzer (Editors)






This document – excluding quotations and otherwise identified parts – is licensed under the Creative Commons Attribution-Share Alike 4.0 International License (CC BY-SA 4.0: <https://creativecommons.org/licenses/by-sa/4.0/>)

Bibliographic information published by Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.dnb.de>.

ISBN 978-3-7376-0945-6

DOI: <https://doi.org/doi:10.17170/kobra-202103173535>

© 2021, kassel university press, Kassel
<https://kup.uni-kassel.de>

kassel
university 
press

Printing Shop: Print Management Logistics Solutions, Kassel
Printed in Germany

Preface

The *Organic Computing Doctoral Dissertation Colloquium* (OC-DDC) series [8] is part of the Organic Computing (OC) initiative which is steered by a Special Interest Group of the German Society of Computer Science (Gesellschaft für Informatik e.V.). It provides an environment for PhD students who have their research focus within the broader OC [10] community to discuss their PhD project and current trends in the corresponding research areas, including, for instance, research in the context of Autonomic Computing [5], Self-adaptive (Software) Systems [7], ProActive Computing [14], Complex Adaptive Systems [4], Self-Organisation [17], Self-Aware Computing [6], Interwoven Systems [16], and related topics.

Organic Computing postulates to equip technical systems with 'life-like' properties. Technically, this means to move traditional design-time decisions to runtime and into the responsibility of systems themselves. As a result, systems have a dramatically increased decision freedom that leads to highly autonomous behaviour [20]. The goal of this process is to allow for self-adaptation and self-improvement of system behaviour at runtime [19]. Especially since conditions that occur at runtime can only be anticipated to a certain degree, efficient mechanisms are needed that guide the system's behaviour even in cases of missing knowledge or uncertain environmental status. Consequently, Organic Computing summarises a variety of aspects and techniques that are needed to finally develop such mechanisms [18].

The main goal of the colloquium is to foster excellence in OC-related research by providing feedback and advice that are particularly relevant to the PhD students' studies and career development. Thereby, participants are involved in all stages of the colloquium organisation and the review process to gain experience. Consequently, they all have been invited to serve in the Programme Committee, being supported by organisers (Christian Krupitzer from University of Hohenheim and Sven Tomforde from University of Kassel) and the members of the advisory board (Christian Müller-Schloer from Leibniz University of Hannover, Jörg Hähner from University of Augsburg, and Bernhard Sick from University of Kassel).

Contributions from all over Germany have been accepted — allowing the authors to present their work. In addition, valuable input has been provided to the OC-DDC by Prof. Anthony Stein from the University of Hohenheim as invited speaker. We

would like to take this opportunity to express our gratitude to all participants and in particular to the invited speaker.

This book presents the results of the OC-DDC 2020. Successful participants have been invited to extend their abstracts submitted to the event towards a full book chapter by taking reviews and feedback received at the event into account. Eleven participants prepared a contribution to this book, helped to perform a sophisticated review process, and finally came up with interesting articles summarising their current work in the context of Organic Computing. Hence, the book also gives an overview of corresponding research activities in the field in Germany for the year 2020. The collection of contributions reflects the diversity of the different aspects of Organic Computing. In the following, we outline the contributions contained in this book.

The first contribution by Eric Hutter extends the Artificial Hormone System (AHS) [1] with a priority-based self-healing mechanism to counter overload situations. In particular, the idea is that — as soon as a critical number of nodes in the AHS fails — these resources will no longer be sufficient to heal the system completely. A task-level priority scheme allows for stopping tasks of low priority in order to free up resources for the most critical tasks. As a result, a graceful degradation is achieved which ensures the system's most important functionality to remain working in such overload situations.

The second contribution by Veronika Lesch continues work presented in [9] and proposes a framework for self-learning adaptation planning through optimisation. The framework contains several layers and incorporates an adaptation planning algorithm, situation-awareness, algorithm selection, learning, and an optimisation component. The idea is that using this as a fundamental basis allows engineers to dynamically select and substitute optimisation techniques and their parametrisation at runtime — resulting in a more efficient behaviour characterised by an improved performance.

The third contribution by Martin Neumayer is based on preliminary work on self-organised production systems [12]. It introduces realistic product descriptions, extends the framework to allow for dynamic scheduling in self-organising production systems and aims at investigating solutions for deadlock-avoidance handling multiple types of products at once.

The fourth contribution by Michael Pernpeintner proposes to extend the traditional agent/environment model of a multi-agent system (MAS) with a so-called 'governance component'. This component is able to observe publicly available information about agents and environment, which is then used to guide the action spaces of agents. The idea is that such a mechanism can prevent certain environmental transitions.

The fifth contribution by Wenzel Pilar von Pilchau focuses on the learning capability of OC systems — which has been identified as being central for achieving OC properties [15] and SASO systems in general [3]. The paper focuses on extending OC-based reinforcement learning concepts with (mathematical) interpolation to get information of data points in an area where only neighbouring samples are known. This information then serves as a basis to extend the learning technique with a mechanism for experience replay.

The sixth contribution by Simon Reichhuber is also located in the area of machine learning for self-adaptive systems. Based on the initial vision presented in [2], it

refines the Observer/Controller framework of OC [19] by addressing the challenges of how an online learning system adapts its knowledge according to a changing environment. Such new knowledge is modelled as, e.g., arrival of new classes or changing noise functions. In the presented concept, the control mechanism itself is represented by a traditional machine learning model, here realised as a static model (Support Vector Machine, Decision Tree) which is retrained step-wise and an Online learning model (learning classifier system, Q-learning) that continuously improves its performance.

The seventh contribution by Jens Sager applies OC technology to the domain of energy systems. Based on the motivation that the transition from fossil fuels to renewable energy sources is characterised by an increased decentralisation of power generation, he argues that classical control approaches are continuously replaced by decentralised solutions with the primary goal to allow for better scalability and adaptability. However, grid stability is still a system-wide goal that is to a certain degree in conflict with decentralisation. His work aims at a verification system for open, heterogeneous, stochastic systems by extending existing stochastic contract approaches with reasoning about a changing number of components. This is proposed to be done by the use of an extension of stochastic signal temporal logic contracts to establish relationships between variables on different system levels.

The eighth contribution by Helena Stegherr focuses on the online optimisation capability of OC systems. Since the number of nature-inspired meta-heuristics for optimisation is still growing, she proposes an adaptive framework for testing and evaluation of these meta-heuristics. To analyse functional components of meta-heuristics with respect to their effects on optimisation, each meta-heuristic is modelled in terms of these components. The underlying idea is that this approach facilitates the detection of general concepts as well as novelties in meta-heuristics, since components can be abstracted and compared.

The ninth contribution by Ingo Thomsen is based on preliminary work on the Organic Traffic Control (OTC) system — a self-adaptive, self-organising, and self-improving traffic management system [11, 13]. Urban traffic is characterised by fluctuating demands within the network and unforeseen disturbances caused by traffic incidents — which can lead to congestion problems. Therefore, the paper proposes a concept for the detection and cooperative validation of possible incidents, which then serves as basis for optimised network-wide traffic light signalisation and route recommendations.

The tenth contribution by Marvin Züfle deals with the challenge that automation of systems results in huge amounts of data. Currently, these data are mostly stored in databases but never analysed further. Based on the scenario of predicting critical conditions from historical observations, the paper presents the vision of a system model tailored for automatic application to monitoring data for predicting failures and other critical conditions. The system model builds upon the design concept of self-aware computing systems and integrates a meta analysis component for method recommendation.

Finally, the eleventh contribution by Martin Jänicke et al. deals with challenges in the field of activity recognition (AR). AR is typically performed on everyday devices

such as smartphones to estimate the current user status and trigger automated actions according to the user's needs. In this work, a novel combination of modelling sensor data and adapting classification systems is proposed that analyses acceleration time series by means of Hidden Markov Models and uses the characteristics as features for a recognition system. Afterwards, this recognition system applies a classifier that can be adapted to be used with with additional sensors at runtime.

We thank all authors for their contributions and we are looking forward to seeing again very interesting OC research at the next OC-DDC. Further, we thank Norbert Schmitt for his involvement as Web Chair.

Hohenheim, January 2021
Kiel, January 2021

Christian Krupitzer
Sven Tomforde

References

1. Brinkschulte, U., Pacher, M., Von Renteln, A.: An artificial hormone system for self-organizing real-time task allocation in organic middleware. In: *Organic Computing*, pp. 261–283. Springer (2009)
2. Calma, A., Kottke, D., Sick, B., Tomforde, S.: Learning to learn: Dynamic runtime exploitation of various knowledge sources and machine learning paradigms. In: *2nd IEEE International Workshops on Foundations and Applications of Self* Systems, FAS*W@SASO/ICCAC 2017, Tucson, AZ, USA, September 18-22, 2017*. pp. 109–116 (2017)
3. D’Angelo, M., Gerasimou, S., Ghahremani, S., Grohmann, J., Nunes, I., Pournaras, E., Tomforde, S.: On learning in collective self-adaptive systems: state of practice and a 3d framework. In: *Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*. pp. 13–24 (2019)
4. Holland, J.H.: Complex adaptive systems. *Daedalus* 121(1), 17–30 (1992)
5. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* 36(1), 41–50 (2003)
6. Kounev, S., Lewis, P., Bellman, K.L., Bencomo, N., Camara, J., Diaconescu, A., Esterle, L., Geihs, K., Giese, H., Götz, S., et al.: The notion of self-aware computing. In: *Self-Aware Computing Systems*, pp. 3–16. Springer (2017)
7. Krupitzer, C., Roth, F.M., VanSyckel, S., Schiele, G., Becker, C.: A Survey on Engineering Approaches for Self-Adaptive Systems. *PMCIJ* 17(Part B), 184–206 (2015)
8. Krupitzer, C., Tomforde, S.: The organic computing doctoral dissertation colloquium: Status and overview in 2019. In: *49. Jahrestagung der Gesellschaft für Informatik, 50 Jahre Gesellschaft für Informatik - Informatik für Gesellschaft, INFORMATIK 2019 - Workshops, Kassel, Germany, September 23-26, 2019*. pp. 545–554 (2019)
9. Lesch, V., Krupitzer, C., Tomforde, S.: Emerging self-integration through coordination of autonomous adaptive systems. In: *IEEE 4th International Workshops on Foundations and Applications of Self* Systems, FAS*W@SASO/ICCAC 2019, Umea, Sweden, June 16-20, 2019*. pp. 6–9 (2019)
10. Müller-Schloer, C., Tomforde, S.: *Organic Computing - Technical Systems for Survival in the Real World*. Birkhäuser (2017)
11. Prothmann, H., Tomforde, S., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Organic traffic control. In: *Organic Computing - A Paradigm Shift for Complex Systems*, pp. 431–446 (2011)
12. Seebach, H., Nafz, F., Steghöfer, J.P., Reif, W.: A software engineering guideline for self-organizing resource-flow systems. In: *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. pp. 194–203. IEEE (2010)
13. Sommer, M., Tomforde, S., Hähner, J.: An organic computing approach to resilient traffic management. In: *Autonomic Road Transport Support Systems*, pp. 113–130 (2016)
14. Tennenhouse, D.: Proactive computing. *Communications of the ACM* 43(5), 43–50 (2000)
15. Tomforde, S., Brameshuber, A., Hähner, J., Müller-Schloer, C.: 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*. pp. 1628–1635 (2011)
16. Tomforde, S., Hähner, J., Sick, B.: Interwoven systems. *Inform. Spektrum* 37(5), 483–487 (2014)

17. Tomforde, S., Kantert, J., Sick, B.: Measuring self-organisation at runtime - A quantification method based on divergence measures. In: Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 1, Porto, Portugal, February 24-26, 2017. pp. 96–106 (2017)
18. Tomforde, S., Müller-Schloer, C.: Incremental design of adaptive systems. *J. Ambient Intell. Smart Environ.* 6(2), 179–198 (2014)
19. Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., Schmeck, H.: Observation and control of organic systems. In: *Organic Computing - A Paradigm Shift for Complex Systems*, pp. 325–338 (2011)
20. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic computing in the spotlight. *CoRR* abs/1701.08125 (2017)

Invited Keynote

An Organic Computing Approach to Resilient Traffic Management

Abstract. All over the globe, urban traffic control systems are characterised by ever increasing congestion effects and negative impact of motorised transport on the environmental conditions. Besides a shift towards alternative transportation systems, an optimisation of today's road network and a better utilisation of the existing infrastructure play a key role in developing countermeasures for these challenges. Inherent properties such as seasonal effects, time-dependent behaviour, and a local distribution of the control problem result in traffic control being an ideal application domain for techniques from the field of self-adaptive and self-organised systems. In this talk, the Organic Traffic Control (OTC) system [2, 3] is introduced, which applies Organic Computing [1] techniques such as the generalised Observer/Controller pattern [5], continuous self-adaptation, and reinforcement learning under safety considerations [4] to the control of traffic signals. Based on the basic OTC system, extensions for establishing decentralised progressive signal systems are presented [6, 7]. The system is evaluated in close-to-reality simulations of road networks situated in Hamburg, Germany and analysed by means of traffic domain specific metrics such as averaged travel times, number of stops, or emissions.

Sven Tomforde
Intelligent Embedded Systems Lab, University of Kassel

References

1. Müller-Schloer, C., Tomforde, S.: Organic Computing – Technical Systems for Survival in the Real World. *Autonomic Systems*, Birkhäuser Verlag (October 2017), ISBN: 978-3-319-68476-5
2. Prothmann, H., Tomforde, S., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Organic Traffic Control. In: *Organic Computing – A Paradigm Shift for Complex Systems*, pp. 431 – 446. *Autonomic Systems*, Birkhäuser Verlag, Basel, CH (2011)
3. Sommer, M., Tomforde, S., Hähner, J.: Resilient Traffic Management with Organic Computing Techniques. In: *Proceedings of the 1st International Systems Competition on Autonomic Features and Technologies for Road Traffic Modelling and Control Systems*, held together with The 16th International IEEE Conference on Intelligent Transport Systems (IEEE-ITS13) at Den Hague, Netherlands (2013), (Winner of the ARTS Competition at IEEE-ITS13)
4. Tomforde, S., Brameshuber, A., Hähner, J., Müller-Schloer, C.: Restricted On-line Learning in Real-world Systems. In: *Proc. of the IEEE Congress on Evolutionary Computation (CEC11)*, held 05 Jun - 08 Jun 2011 in New Orleans, USA. pp. 1628 – 1635. IEEE (2011)
5. Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., Schmeck, H.: Observation and Control of Organic Systems. In: Müller-Schloer, C., Schmeck, H., Ungerer, T. (eds.) *Organic Computing - A Paradigm Shift for Complex Systems*, pp. 325 – 338. *Autonomic Systems*, Birkhäuser Verlag (2011)
6. Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Possibilities and Limitations of Decentralised Traffic Control Systems. In: *2010 IEEE World Congress on Computational Intelligence (IEEE WCCI 2010)*, held 18 Jul - 23 Jul 2010 in Narcelona, Spain. pp. 3298–3306. IEEE (2010)
7. Tomforde, S., Prothmann, H., Rochner, F., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Decentralised Progressive Signal Systems for Organic Traffic Control. In: Brueckner, S., Robertson, P., Bellur, U. (eds.) *Proceedings of the 2nd IEEE International Conference on Self-Adaption and Self-Organization (SASO'08)*, held in Venice, Italy (October 20 - 24, 2008). pp. 413–422. IEEE (2008)

Contents

Part I Contributions from the OC-DDC

1 Degrading Systems by Priority-Based Self-Healing using an Artificial Hormone System	
<i>Eric Hutter</i>	3
2 Towards a Framework for Self-Learning Adaptation Planning through Optimisation	
<i>Veronika Lesch</i>	17
3 Research Challenges in Adaptive Production Systems	
<i>Martin Neumayer</i>	33
4 Self-Learning Governance of Competitive Multi-Agent Systems	
<i>Michael Pernpeintner</i>	47
5 Interpolated Experience Replay - A Roadmap	
<i>Wenzel Pilar von Pilchau</i>	65
6 Opportunistic Knowledge Adaptation in Self-Learning Systems	
<i>Simon Reichhuber</i>	79
7 Verification of Open Stochastic Heterogeneous Systems with Stochastic Contracts	
<i>Jens Sager</i>	93
8 A Framework for a Component-based Comparison of Metaheuristics	
<i>Helena Stegherr</i>	109
9 Incident-aware Resilient Traffic Management for Urban Road Networks	
<i>Ingo Thomsen</i>	125

10 Towards a Self-Aware Prediction of Critical States

Marwin Züfle 139

11 Self-adaptation using discriminative, dynamic models for activity recognition

Martin Jänicke, Vitor Fortes Rey, Bernhard Sick, Sven Tomforde, Paul Lukowicz 157

List of Contributors

Eric Hutter

Eingebettete Systeme
Goethe University Frankfurt
Robert-Mayer-Str. 11-15
60325 Frankfurt, Germany
hutter@es.cs.uni-frankfurt.de

Veronika Lesch

Chair of Software Engineering
University of Würzburg
Am Hubland
97074 Würzburg, Germany
veronika.lesch@uni-wuerzburg.de

Martin Neumayer

Institute for Software & Systems Engineering
University of Augsburg
Universitätsstraße 6a
86159 Augsburg, Germany
veronika.lesch@uni-wuerzburg.de

Michael Pernpeintner

Institute for Enterprise Systems (InES)
University of Mannheim
Schloss
68131 Mannheim, Germany
pernpeintner@es.uni-mannheim.de

Wenzel von Pilchau

Organic Computing Group
Augsburg University
Eichleitnerstr. 30
86159 Augsburg, Germany
wenzel.pillar-von-pilchau@informatik.uni-augsburg.de

Simon Reichhuber

Intelligent Systems
Christian-Albrechts-Universität zu Kiel
Hermann-Rodewald-Str. 3
24118 Kiel, Germany
sir@informatik.uni-kiel.de

Jens Sager

OFFIS - Institute for Information Technology
Escherweg 2
26121 Oldenburg, Germany
jsager@offis.de

Helena Stegherr

Organic Computing Group
Augsburg University
Eichleitnerstraße 30
86159 Augsburg, Germany
helena.stegherr@informatik.uni-augsburg.de

Ingo Thomsen

Intelligent Systems
Christian-Albrechts-Universität zu Kiel
Hermann-Rodewald-Str. 3
24118 Kiel, Germany
int@informatik.uni-kiel.de

Marwin Züfle

Software Engineering Group
University of Würzburg
Am Hubland
97074 Würzburg, Germany
marwin-zuefle@uni-wuerzburg.de

XVIII List of Contributors

Martin Jänicke

Intelligente Eingebettete Systeme
University of Kassel
Willhelmshöher Allee 73
34121 Kassel, Germany
mjaenicke@uni-kassel.de

Vitor Fortes Rey

German Research Center for Artificial
Intelligence
Trippstadter Straße 122
67663 Kaiserslautern, Germany
vitorrey@gmail.com

Prof. Dr. Paul Lukowicz

German Research Center for Artificial
Intelligence

Trippstadter Straße 122
67663 Kaiserslautern, Germany
paul.lukowicz@dfki.de

Prof. Dr. Sven Tomforde

Intelligent Systems
Christian-Albrechts-Universität zu Kiel
Hermann-Rodewald-Str. 3
24118 Kiel, Germany
st@informatik.uni-kiel.de

Prof. Dr. Bernhard Sick

Intelligent Embedded Systems Lab
University of Kassel
Willhelmshöher Allee 73
34121 Kassel, Germany
bsick@uni-kassel.de

Contributions from the OC-DDC

Degrading Systems by Priority-Based Self-Healing using an Artificial Hormone System

Eric Hutter

Goethe University Frankfurt, Frankfurt am Main, Germany
hutter@es.cs.uni-frankfurt.de

Abstract. We outline concepts for resolving overload situations using priority-based self-healing mechanisms in an Artificial Hormone System (AHS). The AHS is a distributed middleware based on Organic Computing principles. It allows to distribute tasks to processing nodes in a self-organising way and has no single-point-of-failure. Node failures can be detected automatically and cause the relocation of any affected tasks to operational nodes, providing self-healing capabilities as long as sufficient computational resources are available.

However, once a certain number of nodes has failed, these resources might no longer be sufficient to heal the system completely. By assigning each task with a priority, it is possible to stop tasks of low priority in order to free up resources for the most critical tasks. This allows graceful degradation and ensures the system's most important functionality remains working in such overload situations. We present a model for overload situations as well as two strategies to realise this kind of self-healing.

Keywords: Artificial Hormone System, Organic Computing, Self-Organisation, Self-Healing, Task Distribution

1.1 Introduction

While embedded systems grow increasingly complex, their components' feature size is shrinking. This increases the probability of failures, e.g. bit flips induced by radiation effects [10]. Thus, ways to simultaneously handle the overall system's complexity while coping with component failures have to be found. The *Artificial Hormone System* (AHS) [2, 11], a distributed middleware based on Organic Computing principles, attempts to achieve these goals. It adapts the biological endocrine system's fundamental principles to technical systems in order to assign tasks to a distributed system's nodes without having a single-point-of-failure. By automatically establishing an initial task distribution, it is self-configuring. Additionally, if one processing node fails, the other nodes automatically re-distribute the affected tasks. Thus, the AHS is also self-healing.

However, these self-healing capabilities only work if the remaining nodes' computational capacities are sufficient for reassigning all failed tasks: If too many nodes

fail, it is undefined which tasks will be running in the system. We thus propose a priority-based extension to the AHS: By assigning a priority to each task, it is possible to systematically degrade the system in such *overload* situations by stopping tasks of low importance in order to keep high-priority tasks running. This can allow a system to maintain at least its most important functionality.

Since the AHS is applicable for use in real-time systems because the time required for self-configuration and self-healing can be bounded, the proposed priority-based extension must also be real-time capable. As a result, bounding the time required to degrade the system in an overload situation is a major concern.

This paper is structured as follows: We first present related work in Section 1.2 and briefly describe the AHS in Section 1.3. Afterwards, we present our priority-based extension as well as our concept for degrading the system in overload situations in Section 1.4. Section 1.5 analyses the time bounds of our degradation scheme and Section 1.6 concludes this paper.

1.2 Related Work

The approach presented in this paper allows a distributed system to recover from hardware failures by means of automatic and dynamic (re)configuration.

Traditionally, the robustness of systems against such failures has often been increased by means of redundancy: By duplicating safety-critical functional units, a redundant unit in a hot stand-by mode can take over in case the primary unit fails. This kind of redundancy is often used for safety-critical electronic control units (ECUs) in the automotive domain. However, recent approaches like the *AutoKonf* project [12] try to reduce costs by using a single backup ECU that is shared between multiple different ECUs. As a result, a single ECU failure can be compensated.

In contrast, our approach is more fine-grained by distributing individual tasks to a distributed system's available computing nodes. By introducing task priorities, recovery from multiple node failures can be achieved by gradually degrading the system. This allows to reduce the number of required backup units while still achieving a high degree of failure tolerance. Our approach is based on the AHS which is introduced in Section 1.3 and uses Organic Computing principles.

Organic Computing (OC) tries to adapt organisation principles observed in biological systems to technical systems. As a result, such OC-based systems exhibit various so-called self-* properties such as self-organisation, self-configuration, self-improvement and self-healing [14]. In general, OC entails a postponement of many traditional design-time decisions into the system's runtime [15].

The dynamics achieved by realizing our system based on OC principles distinguish it from approaches like [5] where a (offline) pre-computed adaptation scenario is applied when nodes fail or an overload situation occurs. Similarly, in the *AutoKonf* project, the backup ECU that takes over on failures is also predetermined. In contrast, the AHS allows a more dynamic reaction to system failures: Here, all affected tasks may be relocated independently to possibly different nodes when a failure occurs, all

relocations are determined dynamically and can respect the individual nodes' states (e.g. temperature or power budget).

However, the AHS is by no means the only approach to distribute tasks in distributed systems: [18] deals with an improved Contract Net Protocol for task assignment, also employing self-healing capabilities and task priorities. However, by not being completely decentralised and not guaranteeing hard real-time bounds, this approach's goals differ from ours.

Task distribution has been a research topic in Wireless Sensor Networks (WSNs) for several years, with early uses of self-organization to tackle this problem [3] as well as current ones [17]. More recent work has been using game theory [4] or particle swarm optimization [8] with different notions of task priorities, ranging from deadline-based priorities to ones derived from the task graph, e.g. the number of dependent tasks. However, with WSNs typically having a limited energy budget per node, energy-efficiency is one of the main concerns in these works rather than guaranteeing hard real-time behaviour as in our approach.

1.3 The Artificial Hormone System

As mentioned before, the *Artificial Hormone System (AHS)* is a decentralised middleware to assign tasks in a distributed system. It works by realising control loops based on short digital messages, called *hormones*, on all *processing elements (PEs)* in the system. *Eager values* are exchanged for all tasks and indicate a PE's suitability for a task. In every cycle of the hormone control loop (called a *hormone cycle*), each PE tries to make a decision upon one task. This is done by comparing its own eager value with all received eager values. If it has sent the highest eager value for some task *T* in the current cycle, it has won *T* and may start executing it.

A PE executing a task sends out an additional hormone, called the acquisition *suppressor*. When received by other PEs, this suppressor will reduce their eager values for this specific task, preventing them from taking additional instances of this task and thus stabilising the system.

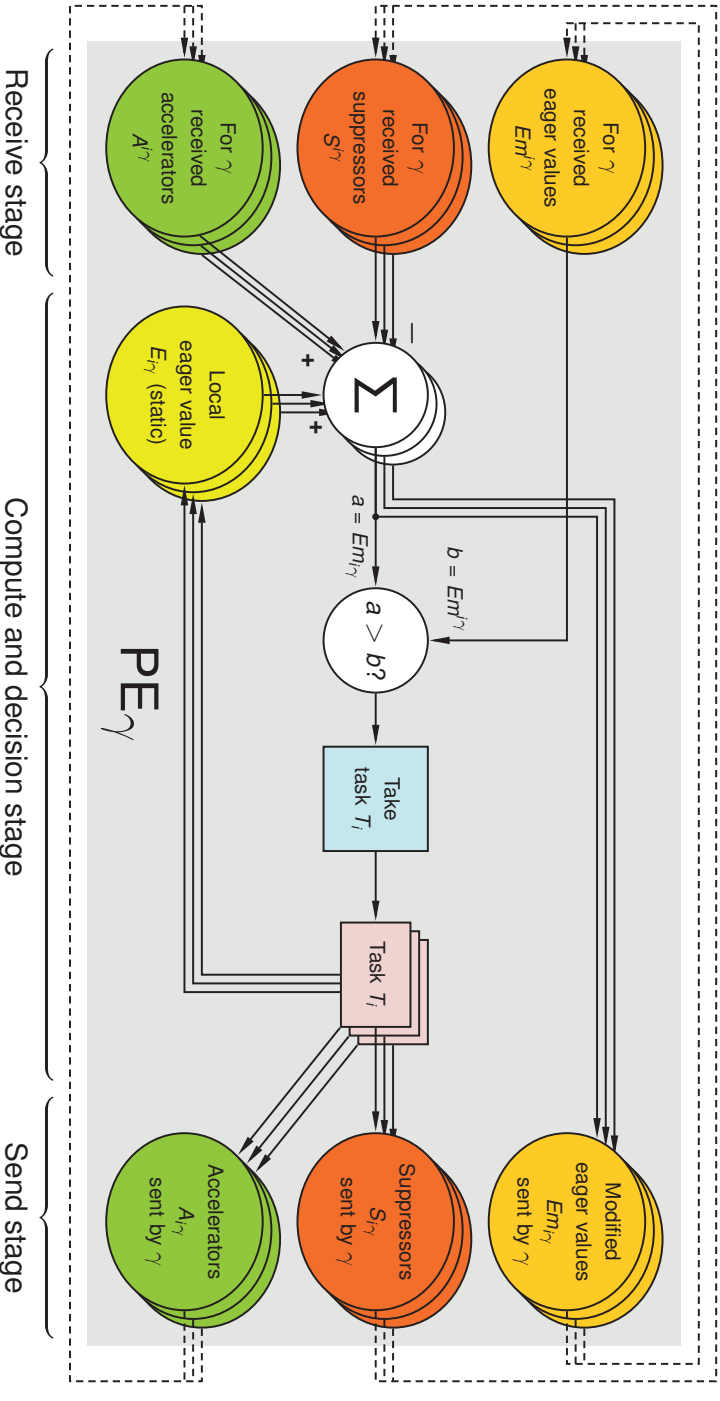
A third class of hormones, the *accelerators*, act antagonistic to the suppressors and increase a task's eager value in order to form functional clusters of related tasks on neighbouring PEs.

Figure 1.1 shows the AHS' hormone control loop that distributes tasks among PEs by exchanging the aforementioned types of hormones.

The AHS automatically distributes all tasks to the available PEs: Once all tasks are distributed, the acquisition suppressors lower all sent eager values to 0 and thus no additional instances are taken. Therefore, by finding an initial task distribution by itself at run-time, the AHS is *self-configuring*.

In addition, by being entirely decentralised, the AHS has no single-point-of-failure: If a PE fails, the remaining PEs will notice this failure since no suppressor hormones will be received by the failed PE any longer. Thus, the affected tasks' eager values rise above 0 again and the tasks will be re-assigned to the available PEs. Therefore,

Fig. 1.1. The AHS' hormone control loop. By exchanging hormones, the PEs distribute the available tasks among themselves in a self-organising way: Each PE calculates and broadcasts a modified eager value for each task. If a PE's sent modified eager value for some task T_i is greater than all received eager values for T_i , it may take this task. In turn, the PE will spread additional instances of T_i and accelerators to neighbouring PEs (to further the formation of clusters of related tasks on those PEs) which are subtracted from resp. added to the other PEs' static eager values for the tasks they are appropriate for.



by re-configuring the system, the AHS compensates this PE failure and can thus be regarded as *self-healing*.

Furthermore, hard time bounds for the initial self-configuration as well as self-healing can be guaranteed, thus making the AHS suitable for use in real-time systems: The self-configuration needs at most m hormone cycles to distribute m different tasks among the PEs while self-healing takes no longer than $m_f + a$ hormone cycles, where m_f is the number of failed tasks to be redistributed and a is a constant describing the time required to notice the PE failure in the first place [1]. For the AHS' current implementation, $a = 2$ hormone cycles holds.

For more detailed information on the AHS and its time bounds, please refer to [1, 2, 16].

1.4 Priority-Based Task Distribution

Our priority-based extension to the AHS works by monitoring the hormones exchanged in the system: Each PE postpones the allocation of some task T if the received eager values suggest that another PE *might* allocate a task T' having a higher priority than T .

In order to do so, the *compute and decision stage* shown in Figure 1.1 has been modified to decide on all tasks in *descending* order of priority. In [9], we have argued that this priority-based AHS takes at most $2m - 1$ hormone cycles for self-configuration of m tasks. This is worse than the m cycles guaranteed by the original AHS but still linear in the number of tasks. Yet, the original AHS neither supports task priorities nor guarantees any order of task assignment. In contrast, the priority-based AHS guarantees that tasks of priority $p' > p$ will be assigned before tasks of priority p , only the order of task assignment within each priority level is nondeterministic (as it can be influenced by environmental conditions, e.g. PE temperature).

1.4.1 Detection of Overload Situations

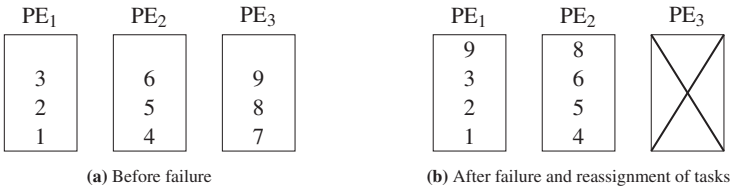


Fig. 1.2. Failure of PE₃ leads to overload situation

If the PEs' combined computational capacities are not sufficient to execute *all* tasks, only the most important ones (based on their high priorities) will be assigned

during self-configuration: Once enough high-priority tasks have been assigned, no PE will have remaining computational resources available and no further task can be assigned in the system.

However, if the failure of a PE leads to such an overload situation, simply relying on the deterministic order of task assignment is not sufficient. This is illustrated in Figure 1.2: Here, each PE can execute at most four tasks. Each number i represents a task of priority i running on a PE. Thus, when PE₃ fails, the high-priority tasks 9, 8 and 7 have to be reassigned on PE₁ and PE₂. Using the mentioned strategy alone, this would result in tasks 9 and 8 being taken by the remaining PEs, but task 7 could not be assigned since both PEs already reached their maximum capacity. Ideally, task 1 would be stopped in order to free up capacities for task 7, but in order to do so, the overload situation has to be detected first.

The AHS models system load in terms of special *load suppressors* that each PE sends to *itself*, thus lowering the modified eager values it broadcasts and effectively limiting the number of tasks it can take. As a result, overload situations in which more tasks exist than can be taken in the system can be recognised by monitoring the exchanged hormones:¹ If, for some task T ,

- a) Eager values but no suppressors are received: T is not assigned because the system is currently self-configuring or self-healing. However, T *could* be assigned by some PE in this cycle.
- b) Suppressors, but no eager values are received: T is assigned to the PE that sent the suppressor.
- c) Neither eager values nor suppressors are received: Currently, no PE can execute T and thus the system is in an overload situation.

If the system is determined to be in an overload situation but some high-priority task T is not running, low-priority tasks have to be stopped in order to allow T 's assignment. This has to be done in a distributed manner, ensuring that it works reliably and the total time required to bring the system into a well-defined (but degraded) state must be bounded to allow meeting real-time requirements.

1.4.2 Analysis of Overload Situations

In order to formally analyse overload situations and compare different strategies to quickly degrade the system, we propose the following model:

- PE _{x} fails and PE₁ . . . PE _{v} remain operational.
- All tasks induce equal load on each PE and each PE is capable to execute any task.
- Each PE can execute at most m tasks.
- At the instant PE _{x} fails, all PEs are executing exactly m tasks.

¹ In order to reduce the amount of communication required, hormones with value zero are generally not sent. Thus, not receiving any modified eager values for some task T means that *each* PE “sent” a modified eager value of zero.

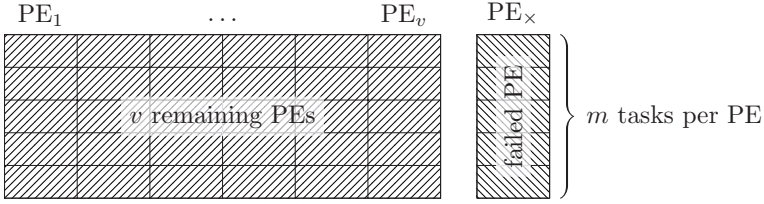


Fig. 1.3. Visualization of overload model

Figure 1.3 visualises this model. In addition, we make the following fundamental assumption:

Assumption 1 Any task may be (temporarily) stopped at any time in order to free up resources for allocation of some task with higher priority.

Using the AHS and utilising this model, the time required to self-heal the system in an overload situation is dependent on the number of tasks that need to be stopped as each task stop will eventually be followed by the (re-)assignment of a (possibly different) task. Thus, it is of great importance to reliably stop as few tasks as possible. Since calculating the optimal set of tasks to stop would require additional communication between the PEs (and thus time), this is generally not feasible and an approximation is necessary. We thus propose two strategies to stop tasks:

Strategy 1 Let T be the highest-priority task that is not currently running. Upon noticing an overload situation, each PE shall stop all running tasks whose priority is lower than T 's priority.

Strategy 2 Let T be the highest-priority task that is not currently running. Upon noticing an overload situation, each PE shall stop its lowest-priority running task if its priority is lower than T 's priority.

We call Strategy 1 *naive task dropping* since it is arguably the simplest strategy. In contrast, Strategy 2 is called *eager task dropping* since dropping one task per PE ensures its eager values rise above zero in the next hormone cycle for every task that is not assigned to any PE.

1.4.2.1 Example

In order to show an example of how these strategies differ, consider the scenario with $m = 3$ and $v = 3$ as shown in Figure 1.4. Task 12 is the highest-priority task that is not currently running. Using Strategy 1, $PE_1 \dots PE_3$ would thus immediately stop *all* $mv = 9$ running tasks. Afterwards, tasks 12, \dots , 4 would be assigned, completing the self-healing.

In contrast, Strategy 2 would proceed in multiple phases, each characterised by stopping k tasks, then followed by assigning the k highest-priority tasks. This is shown in Figure 1.5:

PE ₁	PE ₂	PE ₃	PE _×
7	4	1	10
8	5	2	11
9	6	3	12

Fig. 1.4. Example scenario with $v = 3$ and $m = 3$

PE ₁	PE ₂	PE ₃		PE ₁	PE ₂	PE ₃		PE ₁	PE ₂	PE ₃	
12	11	10	1	12	11	10	1	12	11	10	1
8	5	2	4	8	7	5	2	8	7	5	2
9	6	3	7	9	6	3	4	9	6	4	3

(a) After first phase (b) After second phase (c) After third phase

Fig. 1.5. Degrading the system using Strategy 2. The rightmost column shows the non-running tasks.

Phase 1: Tasks 1, 4 and 7 are stopped. Afterwards, tasks 12, 11 and 10 are assigned.

Phase 2: Tasks 5 and 2 are stopped. Afterwards, task 7 is assigned and task 5 is re-assigned.

Phase 3: Task 3 is stopped. Afterwards, task 4 is re-assigned.

After the third phase, only the $mv = 9$ highest-priority tasks are running and the self-healing is finished. As a result, only six tasks have been stopped in contrast to Strategy 1 which stopped nine tasks.

1.5 Analysing the Naive Task Dropping Strategy

In the following, we analyse Strategy 1 in more detail.

1.5.1 Worst Case

Obviously, the worst case has already been shown in the example above: If the system’s highest-priority task was running on PE_×, all $m \cdot v$ running tasks will be stopped. With a hormone cycles required to notice the failure of PE_×, one hormone cycle to drop mv tasks and send positive eager values again as well as $2 \cdot (mv) -$

1 hormone cycles to assign the mv highest-priority tasks (cf. Section 1.4), the total time required to bring the system into a stable state is bounded by $2mv + a$ hormone cycles.

1.5.2 Average Case

As we have seen, the strategy's worst case is particularly bad. Although the time required for self-healing the system in an overload situation can be bounded, a strategy that stops as few tasks as possible would still be favourable for (hard) real-time applications. Consequently, the naive task dropping strategy is no good choice for real-time applications.

However, should this worst case primarily be due to some rare outlier configurations, the strategy might in fact be suited for use in applications without real-time requirements despite its bad worst case behaviour. We thus want to look at the average number of tasks that are stopped by Strategy 1: Should the expected number of task stops be significantly lower than the worst case ($m \cdot v$ tasks), the strategy might still be useful for certain applications.

The following theorem allows to calculate the expected number of task stops:

Theorem 1. *Let all $m \cdot (v + 1)$ tasks be distributed randomly among $PE_1 \dots PE_v$ and PE_x . Furthermore, let X be a random variable that represents the number of tasks stopped by Strategy 1 and $\mathbf{E}[X]$ its expected value. Then,*

$$\mathbf{E}[X] = \frac{m^2 v}{1 + m}$$

holds.

In [9], we have proven Theorem 1 by directly calculating the probability distribution of X and deriving its expected value. However, we want to give an elegant alternative proof of this result in the following.

1.5.2.1 Background: Number of Exceedances

In order to do so, we introduce the *number of exceedances* as defined in [6]:

Definition 1 (Number of exceedances, continuous case). *Given two samples S_1, S_2 of sizes n and N whose elements are sampled independently from the same continuous distribution, let ξ_i ($1 \leq i \leq n$) be the i -largest observation in the first sample S_1 of size n .*

The number of exceedances Y_i shall now be defined as the number of observations in the second sample S_2 that are equal to or exceed ξ_i :

$$Y_i := |\{s \in S_2 \mid s \geq \xi_i\}|.$$

[13] generalises this definition to discrete probability distributions:

Definition 2 (Number of exceedances, discrete case). Let S_1 be a sample of n balls that are drawn randomly and without replacement from an urn that contains L balls numbered with different real numbers. Let S_2 be a sample of N balls that are drawn afterwards (also without replacement) from the remaining $L - n$ balls ($L \geq n + N$). Furthermore, let ξ_i ($1 \leq i \leq n$) be the i -largest number in S_1 .

The number of exceedances Y_i shall now be defined as the count of numbers in the second sample S_2 that exceed ξ_i :²

$$Y_i := |\{s \in S_2 \mid s > \xi_i\}|.$$

Interestingly, the distribution of the number of exceedances Y_i is identical in both cases [13]. In particular, its expected value can be calculated as follows:

Lemma 1. Let $\mathbf{E}[Y_i]$ denote the expected value of Y_i . Then,

$$\mathbf{E}[Y_i] = N \cdot \frac{i}{n+1}$$

holds.

Proof. See [6]. An alternative proof can be found in [7].

1.5.2.2 Application to the Naive Task Dropping Strategy

In order to prove Theorem 1, we will now model the number of tasks stopped by Strategy 1 in terms of the *number of exceedances* in the discrete case. Since our model assumes all priorities to be different, we can consider the distribution of $m \cdot (v + 1)$ different tasks to $v + 1$ PEs as an urn experiment in which, for each of the m failed tasks on PE_\times , a priority is drawn without replacement from an urn containing the priorities $1, \dots, m \cdot (v + 1)$. Afterwards, for each of the $m \cdot v$ tasks on the remaining $\text{PE}_1 \dots \text{PE}_v$, a priority is drawn without replacement from the remaining $m \cdot v$ priorities in the urn.

As illustrated in Figure 1.6a, the number of tasks dropped by the naive task dropping strategy is now equal to the number of priorities in the second sample that are *lower* than the *highest* priority in the first sample.

If we draw *negative* priorities instead (Figure 1.6b), the tasks stopped by the strategy are all tasks whose priority is *higher* than the *lowest* priority running on PE_\times . These, however, are the tasks from the second sample that *exceed* the m -highest priority ξ_m from the first sample and thus the number of task stops equals the number of exceedances Y_m . This finally allows to prove Theorem 1 as follows:

Proof. As argued, $X = Y_m$ holds. With the first sample consisting of $n = m$ elements and the second sample consisting of $N = m \cdot v$ elements,

$$\mathbf{E}[X] = \mathbf{E}[Y_m] = N \cdot \frac{m}{n+1} = m \cdot v \cdot \frac{m}{m+1} = \frac{m^2 v}{1+m}$$

holds per Lemma 1. \square

² Since the samples are drawn without replacement and all balls are numbered with different numbers, no element in the second sample can be equal to ξ_i .

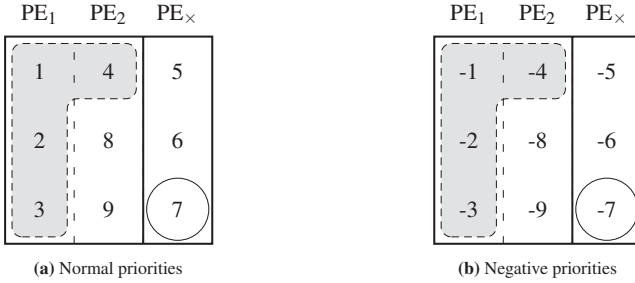


Fig. 1.6. Example: Number of exceedances applied to naive task dropping strategy. The marked tasks will be stopped by the strategy.

1.5.2.3 Discussion

When comparing the expected number of task stops as given by Theorem 1 with the worst case of $m \cdot v$ task stops (cf. Section 1.5.1), it becomes obvious that the naive task dropping strategy does not perform significantly better on average than its worst case. Especially for large values of m , the average number of task stops is approximately equal to the worst case as the limit of their quotient approaches one:

$$\lim_{m \rightarrow \infty} \frac{\text{worst case}}{\text{average case}} = \lim_{m \rightarrow \infty} \frac{mv}{\frac{m^2 v}{1+m}} = \lim_{m \rightarrow \infty} \left(1 + \frac{1}{m}\right) = 1.$$

As argued before, the naive task dropping strategy is not an optimal choice for real-time systems. Furthermore, Theorem 1 shows that it also does not perform substantially better on average which also limits its applicability to systems with no hard real-time bounds. Nevertheless, these results may serve as an important baseline that more sophisticated task dropping strategies will have to be compared to in order to evaluate their usefulness. For instance, the example in Section 1.4.2 has already shown a scenario in which Strategy 2 performed better than the naive task dropping strategy. Future work will deal with bounding this strategy's number of task stops and comparing it to Strategy 1.

1.6 Conclusion and Outlook

We gave an overview about the basic ideas behind a priority-based extension to the Artificial Hormone System. By utilising task priorities, it is possible to degrade the system if too many PEs fail to execute all tasks. This is a situation not handled by the original AHS: By being oblivious to task priorities, the actual tasks executed in such overload situations would be chosen non-deterministically.

We presented a model for the analysis of such overload situations and two strategies to degrade the system. While the first one naively stops all tasks whose priority is lower than the highest priority of all failed tasks, the second strategy stops

one task per PE and replaces it with a high-priority task until the system is correctly degraded.

We have analysed the first strategy with regard to its worst and average cases: In the worst case, *all* running tasks are stopped and it does not perform substantially better on average, either.

In the future, we plan to bound the number of tasks stopped by the second strategy and compare it to the first one, determining situations in which one strategy is superior to the other. Additionally, we plan to develop more elaborate strategies that guarantee even fewer task stops.


References

1. Brinkschulte, U., Pacher, M.: An Agressive Strategy for an Artificial Hormone System to Minimize the Task Allocation Time. In: 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops. pp. 188–195. IEEE, Shenzhen, China (Apr 2012)
2. Brinkschulte, U., Pacher, M., von Renteln, A., Betting, B.: Organic Real-Time Middleware. In: Higuera-Toledano, M.T., Brinkschulte, U., Rettberg, A. (eds.) Self-Organization in Embedded Real-Time Systems, pp. 179–208. Springer New York, New York, NY (2013)
3. Dressler, F., Krüger, B., Fuchs, G., German, R.: Self-Organization in Sensor Networks using Bio-Inspired Mechanisms. In: Brinkschulte, U., Becker, J., Fey, D., Hochberger, C., Martinetz, T., Müller-Schloer, C., Schmeck, H., Ungerer, T., Würtz, R.P. (eds.) 18th International Conference on Architecture of Computing Systems, Workshops, Innsbruck, Austria, March 2005. pp. 139–144. VDE Verlag (2005)
4. Edalat, N., Tham, C.K., Xiao, W.: An auction-based strategy for distributed task allocation in wireless sensor networks. *Computer Communications* 35(8), 916–928 (May 2012)
5. Fohler, G., Gala, G., Pérez, Daniel, G., Claire, Pagetti: Evaluation of DREAMS resource management solutions on a mixed-critical demonstrator. In: ERTS 2018. 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), Toulouse, France (Jan 2018)
6. Gumbel, E.J., von Schelling, H.: The Distribution of the Number of Exceedances. *The Annals of Mathematical Statistics* 21(2), 247–262 (Jun 1950)
7. Gumbel, E.J.: Elementare Ableitung der Momente für die Zahl der Überschreitungen. *Mitteilungsblatt für Mathematische Statistik* 6, 164–169 (1954)
8. Guo, W., Li, J., Chen, G., Niu, Y., Chen, C.: A PSO-Optimized Real-Time Fault-Tolerant Task Allocation Algorithm in Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems* 26(12), 3236–3249 (Dec 2015)
9. Hutter, E., Brinkschulte, U.: Towards a Priority-Based Task Distribution Strategy for an Artificial Hormone System. In: Brinkmann, A., Karl, W., Lankes, S., Tomforde, S., Pionteck, T., Trinitis, C. (eds.) *Architecture of Computing Systems – ARCS 2020*. vol. 12155, pp. 69–81. Springer International Publishing, Cham (2020)
10. Ibe, E.H.: *Terrestrial Radiation Effects in ULSI Devices and Electronic Systems*. John Wiley & Sons Inc, Singapore (2015)
11. Müller-Schloer, C., Tomforde, S.: *Organic Computing - Technical Systems for Survival in the Real World*. Birkhäuser (2017), <https://doi.org/10.1007/978-3-319-68477-2>

12. Orlov, S., Korte, M., Oszwald, F., Vollmer, P.: Automatically reconfigurable actuator control for reliable autonomous driving functions (AutoKonf). In: 10th International Munich Chassis Symposium 2019: Chassis.Tech Plus (2019)
13. Sarkadi, K.: On the Distribution of the Number of Exceedances. *The Annals of Mathematical Statistics* 28(4), 1021–1023 (Dec 1957)
14. Tomforde, S., Müller-Schloer, C.: Incremental design of adaptive systems. *J. Ambient Intell. Smart Environ.* 6(2), 179–198 (2014), <https://doi.org/10.3233/AIS-140252>
15. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic Computing in the Spotlight. [arXiv:1701.08125 \[cs\]](https://arxiv.org/abs/1701.08125) (Jan 2017)
16. von Renteln, A., Brinkschulte, U., Pacher, M.: The Artificial Hormone System—An Organic Middleware for Self-organising Real-Time Task Allocation. In: Müller-Schloer, C., Schmeck, H., Ungerer, T. (eds.) *Organic Computing — A Paradigm Shift for Complex Systems*, pp. 369–384. Springer Basel, Basel (2011)
17. Yin, X., Dai, W., Li, B., Chang, L., Li, C.: Cooperative task allocation in heterogeneous wireless sensor networks. *International Journal of Distributed Sensor Networks* 13(10) (Oct 2017)
18. Zhang, J., Wang, G., Song, Y.: Task Assignment of the Improved Contract Net Protocol under a Multi-Agent System. *Algorithms* 12(4), 70 (Apr 2019)

Towards a Framework for Self-Learning Adaptation Planning through Optimisation

Veronika Lesch

OrcidID  0000-0001-7481-4099
University of Würzburg, Germany
veronika.lesch@uni-wuerzburg.de
<https://go.uniwiue.de/lesch>

Abstract. The increasing interest in Industry 4.0 and Intelligent Transportation Systems has also increased the interest in so-called Self-adaptive Systems. These systems monitor their environment and the system itself and can react to changes in their highly dynamic environment. Besides Self-adaptive Systems, other research communities exist like Organic Computing, Autonomic Computing, and Self-aware Computing Systems that also focus on transferring responsibilities towards the system itself and reducing the effort at design time. The addressed domains in this paper, i.e., Industry 4.0 and Intelligent Transportation Systems, become more and more complex and increasing digitisation can be observed. Along with the rising demand for automated optimisation in these domains, engineers face difficult challenges, such as deriving the optimal technique and its parametrisation. This work aims to contribute to this development by proposing a framework for self-learning adaptation planning through optimisation. The framework contains several layers and incorporates an adaptation planning algorithm, situation-awareness, algorithm selection, learning, and an optimisation component. By applying this framework, engineers will be able to dynamically select and substitute optimisation techniques and their parametrisation at runtime. Besides, we discuss the framework's applicability in several use cases and derive research questions that need to be addressed.

Keywords: Self-adaptive Systems, Optimisation, Framework, Industry 4.0, Intelligent Transportation Systems.

2.1 Introduction

The latest trend on digitising machines and production processes toward Industry 4.0 and the increasing interest in Intelligent Transportation Systems (ITS) impels academia and industry to research on and develop novel methods for adaptive systems in dynamic environments. The research area of Self-adaptive Systems (SAS) [11] tries to address these challenges. The SAS can change their behaviour and cope with changes in their environment and the system itself. Diverse research communities arose from the trend towards SAS, such as Organic Computing (OC) [17], Autonomic Computing (AC) [8], and Self-aware Computing System (SeAC) [10]. OC, for example, envisions to 'enable future ICT systems to carry out certain tasks on their

own' [17, p.6] and thereby to 'be able to adapt reasonably to changing requirements of their operating environment' [17, p.6]. This definition shows that OC systems try to shift design time decisions towards run time and therefore put the systems themselves into charge. Similar ideas drive the SAS and SeAC communities, as well as several further research directions. Most of these systems incorporate mechanisms or control loops to react to changes in the environment or the system itself and adapt the system's behaviour. A representative of the SAS community is the MAPE-K control loop [8], while the OC community introduced the Observer/Controller concept [25]. The SeAC domain defined the so-called LRA-M loop [10]. As all these concepts aim at the same vision, most of them can be transferred into each other [13]. In this paper, we decided to use the LRA-M loop as it explicitly integrates sensing of phenomena, includes external goals and the interplay of learning and reasoning in such systems.

Besides these SAS, the need for ongoing progress and continuous optimisation in Industry 4.0 and ITS domains raises further challenges for engineers. One example of the Industry 4.0 domain is the optimisation of setup times in a dynamic production environment. Here, multiple machines exist that need to handle several tasks. Each task has a specific required setup time, which can be reduced if tasks of the same type are processed directly after each other. Due to the high complexity of the problem containing machines, tasks, task types, and type-specific setup times, calculating the most efficient production plan is a complex task in itself and therefore requires optimisation algorithms to find a solution in reasonable computing time. Another example of the ITS domain could be vehicle routing. The task of scheduling orders to vehicles and optimising their tours makes vehicle routing an NP-complete problem and, therefore 'one of the most widely studied topics in the field of Operations Research' [3]. As 'there is no single method available for solving all optimisation problems efficiently' [22, p.1], the selection and optimal parameter configuration of these methods is a critical task. Currently, this selection and configuration is performed manually in most cases.

This work aims to contribute to these challenges by introducing the vision of a framework that enables engineers to apply a self-learning optimisation mechanism. This mechanism will be able to dynamically select the required adaptation planning mechanism and tune its parametrisation in accordance with the system's current situation. Further, it learns on an ongoing basis about the executed decisions to improve the decision-making mechanism. The vision of a self-learning optimisation is described by a system model comprised of a layered architecture and an internal control loop based on the LRA-M loop. Additionally, the applicability is discussed in several use cases and results of a first feasibility study are given. Finally, we derive a set of research questions that need to be assessed in the future to achieve the set goals. The contributions of this work are threefold:

- Definition of a framework based on a layered architecture and an adapted LRA-M control loop.
- Discussion of the applicability of the framework based on several use cases from different domains.
- Derivation of research questions to be addressed in future work.

The remainder of this paper is organised as follows. First, Section 2.2 introduces essential background information and discusses related work. Section 2.3 proposes the framework by presenting the system model, the internal control loop, how both are connected, and finally clarifying the assumptions made for this work. The use cases are introduced and discussed in Section 2.4 before Section 2.5 presents the derived research questions. Finally, Section 2.6 concludes the paper.

2.2 Foundations and Related Work

This section introduces background information on optimisation techniques and operations research approaches based on a classification by S. S. Rao [22]. Afterwards, this section gives a brief overview of related work and discusses the delineation of this work.

2.2.1 Foundations on Optimisation Techniques

According to S. S. Rao [22], operations research approaches can be classified into four different categories: (i) mathematical programming or optimisation techniques, (ii) stochastic process techniques, (iii) statistical methods, and (iv) modern or non-traditional optimisation techniques. This work only integrates approaches from the first and the last categories. In the following, some examples of these categories are given. Linear programming is an example of mathematical programming that is part of the first category. ‘Linear programming is an optimisation method applicable for the solution of problems in which the objective function and the constraints appear as linear functions of the decision variables’ [22, p. 119]. Some examples for the last category of modern optimisation techniques and meta-heuristics are local search, genetic algorithms, and ant-colony optimisation. Local search is a method that first calculates a valid solution and then tries to achieve better solutions by exploring a local neighbourhood [7]. Genetic algorithms are based on the natural evolution of a population by using genetics and natural selection [22]. Here, the population size, the parent size, the probability of mutations and many more parameters can be tuned. In every evolution step, the population evolves by reproduction, crossover, and mutation. Ant colony optimisation ‘is based on the behaviour of a colony or swarm of insects, such as ants, termites, bees, and wasps’ [22, p. 708].

2.2.2 Related Work

The problem of reacting dynamically to unforeseen situations, optimising parametrisations of systems as well as comparing the performance of optimisation algorithms in various domains is a highly researched field.

Perrouin *et al.* [19] propose a rule-based approach for meta-self-awareness. Therefore, they use layered MAPE loops to optimise adaptation decisions and to make an adaptive system ‘resilient to a larger number of unexpected situations’ [19]. Fredericks *et al.* [5] present in their work from 2019 an approach that determines the

current situation using clustering. Then, this information is used for optimisation techniques to discover the optimal configuration for black-box systems. A related approach is proposed by Porter *et al.* [20]. They introduce a framework for online learning combined with unsupervised learning to find optimal configurations inside a given search space. Kinner *et al.* [9] propose the idea to re-use knowledge of previous plans for optimisation. They apply a white-box approach with knowledge of the system combined with a genetic algorithm to react to unexpected adaptation scenarios. Further, several authors compare optimisation techniques and analyse their performance. For example, Bischl *et al.* [2] compare model-based single- and multi-objective optimisation techniques for black-box functions. Besides the NSGA-II algorithm, they compare a couple of optimisation algorithms to a Bayesian approach. Another work is from Moreno *et al.* [16]. The authors compare CobRA and PLA—two model predictive control techniques—using the RUBiS benchmark for web and cloud application performance. In their evaluation, the authors focus on the performance of the techniques as well as the needed expertise to use these systems.

Another research direction related to this work is the field of Auto-ML. As the name already indicates, automated machine learning focuses on automating machine learning mechanisms by applying pipelines combined with hyperparameter optimisation to reduce the manual effort. Reinbo, for example, is an Auto-ML framework using task pipelines and implementing reinforcement learning and Bayesian optimisation to determine the parameters automatically [23]. A similar approach is applied by Chai *et al.* in their preprint from 2019, where they propose an Auto-ML framework covering the common machine learning issue of data drift [4]. Thornton *et al.* propose a mechanism to select and optimise hyper-parameters in the context of classification algorithms [24]. Finally, Li *et al.* try to solve the hyper-parameter tuning problem using a random search mechanism in combination with adaptive resource allocation and early-stopping [14].

All the approaches mentioned above already cover parts of our proposed framework, such as a rule-based meta-self-aware approach, situation-awareness, determining the optimal configuration of a system, or a performance comparison of optimisation techniques. However, no other work integrates all these aspects into one framework. The combination of a layered framework with the LRA-M loop and the integration of adaptation planning algorithms, situation-awareness, algorithm selection, learning approaches, and optimisation techniques make the proposed approach unique and a valid contribution to the research community.

2.3 Self-Learning Adaptation Planning Through Optimisation

The increasing interest in SAS, as well as the trend towards Industry 4.0 and Intelligent Transportation Systems (ITS), raise several challenges for optimising these systems. The dynamic structure of these systems and a dynamic environment require an in the same measure flexible possibility to optimise these systems. Therefore, this paper proposes a framework that addresses these challenges by combining a layered architecture with control loop mechanisms from the SeAC area. This section first

introduces the system model consisting of the layered framework, describes the LRA-M loop adaptation, and states assumptions of this work.

2.3.1 System Model

This section introduces the framework to be proposed that is based on a layered architecture. The system model (see Figure 2.1) integrates three layers: (i) Application, (ii) Adaptation Planning, and (iii) Meta-Optimisation. Further, it integrates several components as well as the flow of data for the framework.

The bottom layer of the system model is the application layer ①. Real-world applications from the domain of Industry 4.0 and ITS are considered and discussed in Section 2.4. An assumption for this layer is that all considered systems are digitised and adaptive, that is, to gather and transfer data and that they are able to adapt the system and/or its processes according to given adaptations. The applications monitor themselves as well as their environment and send the data to the next layer.

The middle layer, called adaptation planning ②, includes the algorithm that receives data from the application and uses it to plan adaptations of the system. These algorithms are selected among various existing algorithms and it is not part of the contribution to develop a new algorithm for this layer. The algorithm can range from simple rule-based algorithms over coordination algorithms for platooning to complex (multi-objective) optimisation algorithms (see Section 2.2.1). Further, several algorithms should be provided for each problem statement that are adapted for a specific use case to provide the possibility of algorithm exchanges if necessary. One crucial point for this layer is the monitoring, as performance data of the selected algorithm need to be gathered and transferred— together with the application’s monitoring data— to the next level. This layer receives commands to change the algorithm’s parametrisation or even exchange the algorithm itself.

Finally, the third layer is called meta-optimisation ③. This layer is responsible for optimising the parameters as well as the algorithm selection for layer (2) and therefore integrates four components: (i) situation awareness, (ii) algorithm selection, (iii) learning, and (iv) optimisation. The situation awareness component receives the monitoring data of the application as well as the performance data and categorises the current state of the system. The optimisation component also receives monitoring data and tries to tune the parameters of the adaptation planning algorithm. All this information from situation-awareness and optimisation is given—together with the monitoring and performance data—to the algorithm selection. The algorithm selection uses this information to determine how to tune the algorithm parameters if possible or which algorithm best fits the current situation. This decision, in combination with the monitoring and performance data as well as the determined situation, is given to the learning component. The learning component manages a set of all known situations and according decisions and learns on an ongoing basis which parameter and algorithm combination fits best for the already experienced situations. Further, it is possible to include a fifth component containing forecasting mechanisms to enable a proactive adaptation of the system. Finally, the third layer gives the decisions to the adaptation planning layer that executes them.

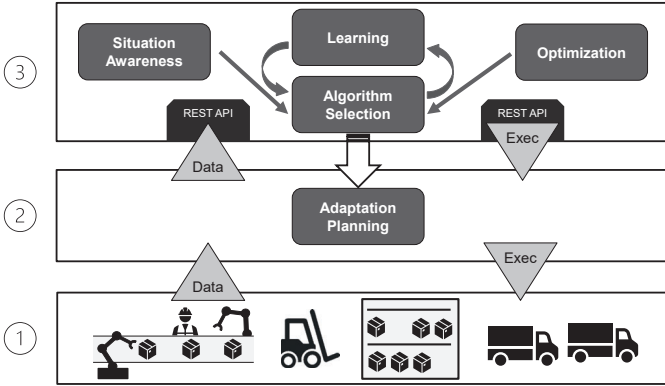


Fig. 2.1. Layered architecture of the framework for self-learning optimisation. Layer 1 (application layer) represents an adaptive system, the adaptation planning called layer is depicted in layer 2 and layer 3 (meta-optimisation) shows the algorithm selection based on situation awareness combined with a learning and an optimisation module.

2.3.2 Adaptation of the LRA-M Loop

The previous section described the general framework. This section explains the control loop used to realise the vision of self-learning adaptation planning using optimisation. The LRA-M loop was first introduced by Kounev *et al.* in 2017 [10] in his work on Self-aware Computing Systems. This loop is quite similar to other concepts like the MAPE-K control loop [8] or the Observer/Controller concept [25] and most of these concepts can be transferred into each other [13]. Thus, we decided to base our control loop on the LRA-M loop and adapt it to match the framework’s requirements. Figure 2.2 depicts the concept of our modified LRA-M loop.

The loop displays the system, also called the self, and the interfaces to its environments. It interacts with the environment by (i) sensing *Phenomena* and storing them as *Empirical Observations*, (ii) receiving *Goals* to be achieved, and (iii) perform *Actions* based on the made decisions. The *Empirical Observations* are sensed in the use case and are required in the *Learn* and *Reason* components. In the ongoing learning process, the observations are abstracted into models that contain knowledge about the environment and the system itself. The learn module contains an *Analysis* component that is able to interpret the observations and updates the models to persist all gathered information. Further, the learning component contains the *Meta-Optimisation* that learns the effects of the actions taken based on the current situation. This enables the system to improve its reasoning and act on an ongoing basis and improve the system’s models and the surrounding environment. These models are used as a basis for the reasoning process that determines actions to be performed as a reaction to a changing environment or to affect the reasoning and learning processes itself. The reason module contains the *Planning by Optimisation* component. It determines actions for the system to adapt to changes in the environment as well to

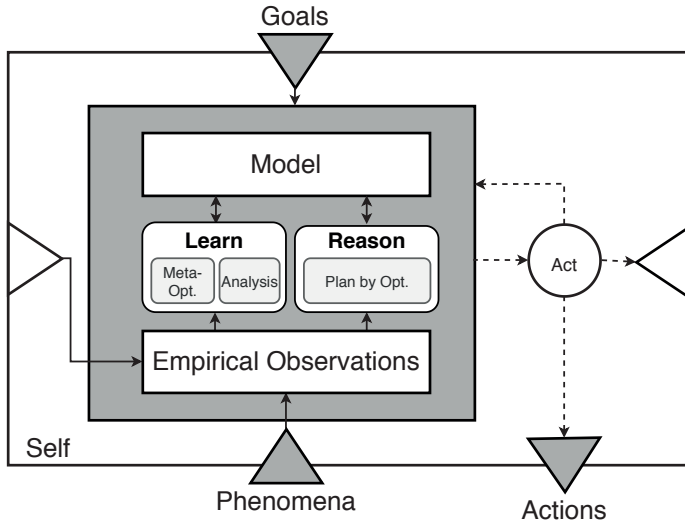


Fig. 2.2. Modified Learn-Reason-Act-Model (LRA-M) Loop based on Kounev *et al.* 2017. The basic LRA-M Loop is extended to contain an analysis and the meta-optimisation in the Learn module and the planning by optimisation step in the Reason module.

adapt the optimisation algorithm and stores all gathered knowledge in the model. The analysis and Meta-optimisation in the learning module, as well as the planning by optimisation component in the reason module, are newly introduced parts and not part of the original definition of the LRA-M loop. These three components build the main contribution in terms of the proposed framework and are meant to be generically applicable to a wide range of suitable use cases.

2.3.3 Mapping the LRA-M Loop to the Layered Architecture

The previous sections described the proposed framework as a rough overview of components in a three-layered view and the analysis and learning perspective of the system by using the adapted LRA-M loop. This section will now present the connection between the two representations by mapping the modules from the loop into the layered architecture. Further, this section gives a first glance at how the framework's generic applicability can be realised.

First of all, the empirical observations, including the phenomena and goals, are sensed in the application layer of the framework as the use case is located there. Further, the decisions of the adaptation planning layer are also part of the sensed phenomena as these are required as additional information sources for the third layer. The meta-optimisation layer of the framework contains the main parts of the LRA-M loop, that is, the learning and reasoning modules. The analysis component inside the learn module includes the situation-awareness and can determine the current situation

based on the gathered observations, for example, by applying clustering algorithms. The plan by optimisation component inside the reason module regards to the algorithm selection component in the third layer as it determines actions for the second layer to adapt to changes in the environment. The meta-optimisation component inside the learn module includes the learning component of the meta-optimisation layer and is responsible for creating models of the current decisions based on received feedback from the application and adaptation planning layers and therefore improve future decisions. This component contains algorithms that optimise the system itself as well as the algorithm selection mechanisms.

Finally, as already mentioned, the framework is meant to be generically applicable to a wide variety of use cases. The Representational State Transfer (REST) APIs will enable this general applicability the framework will provide. On the one hand, a first REST API provides the interface to send all domain-specific data to the framework and give information about the key performance indicators of the lower-level system. Based on this information, the framework will apply the situation analysis, algorithm selection and parameter optimisation. Afterwards, the results of this process are made available via the second REST API, which provides the adaptive system to retrieve the decisions for adapting the used algorithm as well as its parametrisation.

2.3.4 Assumptions

As discussed in the previous section, the applicability in various use cases is one crucial point for the framework. Several assumptions need to be made to realise the framework and to allow for the general applicability.

First of all, it is assumed that the use case is digitised, that is, it can gather and transfer data to a higher-level entity. Further, it utilises an adaptation planning algorithm and adapts the system and/or its processes according to given adaptations. Second, the use case and according adaptation planning work as a standalone system and therefore remain functional regardless of whether the third layer already made a decision. This is especially important when starting the third layer and requesting the first optimised decision. Third, there is no interruption of the two lower levels when the next decision of the third layer needs to be made. The adaptation planner in layer two checks for new decisions from layer three regularly and remains as it is if no decision is currently available. Fourth, the adaptation planner is assumed to be interchangeable, and it may provide the possibility to change its parameters at runtime. Therefore, a managing entity in the second layer is assumed to measure relevant metrics and observations and send this information to the third layer. Fifth, the managing entity in the second layer needs to retrieve adaptation commands from the third layer and executes them. Finally, the framework is designed to handle one adaptation planning entity in the second layer. Here, the amount of managed entities in the first layer is irrelevant for the third layer and it does not make a difference if the adaptation planning manages hundreds of vehicles or only a single production facility with a very limited amount of machines.

2.4 Use Cases

This section discusses several use cases to show the relevance as well as the applicability of the proposed framework. Therefore, use cases from two domains are shown: (i) Industry 4.0 and (ii) Intelligent Transportation Systems. First, the results of a preliminary case study in the domain of Industry 4.0 regarding setup times are presented to show the requirement of an automated mechanism.

2.4.1 Industry 4.0

Optimisation is a central instance for Industry 4.0 as increasing digitisation of all systems and workflows inside a production allows for the automated improvement of all processes and a wide variety of use cases could be realised. However, in this work, we focus on three characteristic use cases from a digitised production environment: setup time reduction, production logistics, and storage assignment and order picking. The first use case is explained in more detail, while the other examples are only briefly outlined.

Setup times. In a flexible production environment, setup times build an important factor for the success of a company as downtimes harm productivity. Hence, the reduction of setup times is a significant part of optimising production processes. However, each production facility has its own characteristics and the decision of the best suitable optimisation mechanism is complicated. Moreover, the flexibility of modern production environments requires adaptation to new situations, for example, the introduction of new products or changes in the various production steps. Thus, the optimisation algorithm needs to be adapted, that is, a reconfiguration or even an exchange of the algorithm might be necessary.

In a first case study, we applied a genetic algorithm to the problem of minimising setup times for a flexible job shop scheduling problem. We implemented the genetic algorithm within the optimisation framework OptaPlanner¹ and defined eight, respectively, four problem-specific crossover and mutator moves. A real-world data set which we received from a German company is used for this study. This data set contains around 600 days during the years 2017 to 2019. However, as we conducted a preliminary study, we selected one random day from the data set. On this day, 14 machines were available to which 85 jobs needed to be scheduled. Ten distinct job types were considered for scheduling. Whenever two similar jobs were executed one after the other, the setup time of the second job could be deleted. We then applied the genetic algorithm on the data of this day and analysed the influence of the parameters that can be tuned to improve the performance of the algorithm. Please refer to Section 2.2.1 for an explanation of the different parameters. We considered a set of possible parameter values summarised in Table 2.1. When analysing the population size, we fixed the other parameters, namely parent size to 50% of the individuals in the population and iteration count to 50. For the analysis of the parent size, both the population size and iteration count were fixed to a value of 50 individuals and 50

¹ We used OptaPlanner with version 7.31.0: <https://www.optaplanner.org/>

Table 2.1. Evaluated parameter values for the genetic algorithm.

Parameter	Possible Values
Population Size	25, 50 , 100, 200
Parent Size	10, 20, 30, 40, 50
Iteration Count	25, 50 , 75, 100

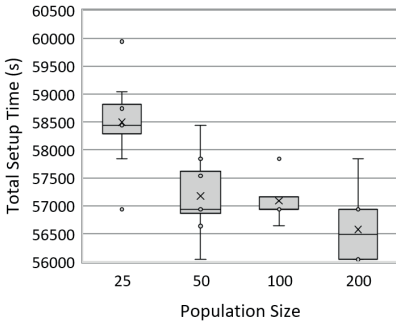


Fig. 2.3. Example evaluation results for parametrising genetic algorithm regarding population size.

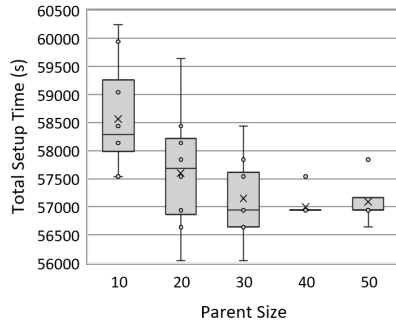


Fig. 2.4. Example evaluation results for parametrising genetic algorithm regarding parent size.

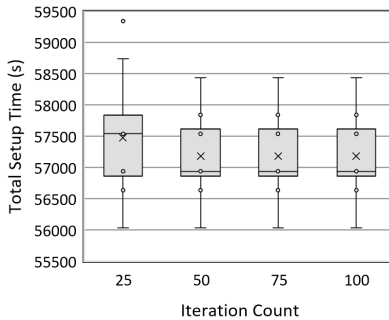


Fig. 2.5. Example evaluation results for parametrising genetic algorithm regarding iteration count.

iterations, respectively. Finally, when analysing the iteration count, the population size was set to 50 individuals and the parent size to 50%. These values are printed in bold in the table. Each experiment was repeated ten times. Figures 2.3, 2.4, and 2.5 show the measurement results for the three parameters population size, parent size, and iteration count, respectively. The first two figures show the influence of the population size and the parent size on the total setup time, which is to be reduced.

Hereby, we ensured that all jobs are fulfilled on the given day by using a hard score in the OptaPlanner framework. In general, an increased value for both parameters has a decreasing effect on the total setup time. However, the effect seems to be less significant for higher parameters, for example, by increasing the parent size from 40 to 50, the total setup time cannot be reduced and can even—in the worst-case—increase the total setup time. In contrast, a change in the iteration count parameter seems not to affect the total setup time. This could be due to the relatively low amount of jobs to be scheduled, which can be optimised within only a few iterations. Therefore, the selected value of 50 iterations per evaluation run seems reasonable, but further tests in the future need to be conducted. Additionally, an evaluation covering the influence of the population size and iteration count on the overall runtime of the algorithm is currently not conducted but planned for the future. All in all, this shows the parametrisation’s fragility and, thus, the importance of carefully determining the parametrisation. Further, possible mutual dependencies need to be analysed and taken into account. Hence, the automated mechanism envisioned in this paper that learns the best configuration for each situation is meaningful in this use case.

Production Logistics. As a second use case, we plan to apply our framework on optimisation problems in production logistics. ‘The fundamental goal of production logistics can be formulated as the pursuance of greater delivery capability and reliability’ [18, p.2]. To achieve this goal, the internal logistic concepts need to be analysed and optimised on an ongoing basis. One part of production logistics can be tugger train systems that take tours in the production facility to provide various materials at different locations [15]. In most cases, these routes are planned statically based on a specific timetable. However, with an increase of the dynamic Industry 4.0 structure of production facilities and the trend towards a batch size of one create new challenges for existing logistics concepts. Therefore, a dynamic planning concept for tugger train systems that is able to adapt to the current situation will be required.

Storage Assignment and Order Picking. In line with the required flexibility of production logistics, storage assignment and order picking face similar challenges when facing digitisation in the domain of Industry 4.0. Currently, storage assignment and order picking are considered separately in most of the literature [26] even if they are strongly coupled [6]. We want to apply our framework in a use case in which we optimise both aspects jointly. This introduces further challenges for our framework, as several interdependent parallel optimisation processes need to be analysed and tuned simultaneously.

2.4.2 Intelligent Transportation Systems

With increasing progress in automotive research and industry, intelligent transportation systems (ITS) arose. In recent years, ITS evolved into ‘an efficient way of improving the performance of transportation system’ [28, p.1] and the availability of data further reinforces the technology. For the ITS domain, we selected two representative use cases to discuss our proposed framework: (i) Vehicle routing and (ii) platooning coordination.

Vehicle Routing. In the literature as well as in industry, vehicle routing is a highly researched field. Braekers *et al.* describe it as ‘one of the most widely studied topics in the field of Operations Research’ [3]. It specifies the task of scheduling orders to vehicles and optimising their tours, making it an NP-complete problem. Therefore, solving the problem using heuristics and optimisation algorithms is mandatory. An example for the Organic Computing domain is given in [21]. However, there is no single vehicle routing problem but many different variants of it, which have different characteristics. Thus, making it meaningful to learn the best working optimisation algorithm and tuning parameters dependent on the current situation.

Platooning Coordination. The increasing trend towards self-driving vehicles in the last years enables new applications such as Platooning. In platooning, vehicles that communicate with each other drive with small inter-vehicle distances [1]. One step further, platooning coordination is the act of determining a platoon to join. Here, various objectives can be taken into account, such as desired driving speed and user comfort [27]. In literature, various coordination mechanisms are proposed, each focusing on different objectives and with diverse restrictions. This diversity in coordination mechanisms and optimisation objectives provides a multifaceted basis to apply the proposed framework for self-learning adaptation planning using optimisation. Platooning coordination and its’ optimisation can be done both centralised and decentralised [13]. However, in this work, we focus on a centralised optimisation of the coordination mechanism. Since the leader of a platoon profits the least of the platooning process, fairness mechanisms should also be integrated into the coordination process. In [12], we proposed a taxonomy of fairness mechanisms and several mechanisms to rotate the leader of a platoon to split negative effects equally among all vehicles inside a platoon.

2.5 Future Research Challenges

This section summarises a set of research questions that arose during the conceptualisation of the framework. These research questions need to be addressed in the future and will guide the research of this work. The questions are structured into three groups: (i) Approach, (ii) Implementation, and (iii) Evaluation.

1. **Approach** - How can all proposed ideas be integrated into one framework?
 - a) Which metrics need to be reported to assess the performance of the adaptation planning algorithm?
 - b) Which monitoring data is required to identify a situation in which the system is currently running?
 - c) How to combine information from the application, the optimisation algorithm’s performance data, and knowledge about the current situation to reason about the best fitting algorithm and / or its configuration?
 - d) How can learning be integrated?
 - e) Which learning techniques fit this approach and the targeted use cases?

- f) Is a justification of the modifications to the system operator required?
2. **Implementation** - Which challenges need to be faced when implementing the approach?
 - a) Which strategies and techniques can be applied to exchange an optimisation algorithm and / or its' parametrisation at runtime?
 - b) How to integrate multiple objectives into the meta-optimisation?
 - c) How can interdependent optimisation problems be parallelly analysed?
 - d) How can the required actions be communicated to the application?
 3. **Evaluation** - How to evaluate the meta-optimisation framework?
 - a) Which metrics are required to assess the performance of the framework?
 - b) Which data sets can be used to evaluate the framework?
 - c) Which use cases and scenarios provide a meaningful evaluation?
 - d) How transferable are the found results to other use cases?

2.6 Conclusion

The latest trends toward Industry 4.0 and Intelligent Transportation Systems have also increased the interest in Self-adaptive Systems (SAS) as they operate in a highly dynamic environment and can adapt to these changes. Besides these SAS, other research communities with a similar vision arose, such as Organic Computing, Autonomic Computing, and Self-aware Computing Systems. All communities proposed concepts that support the development of dynamic and intelligent systems. Further, the requirement to continuously improve a system and optimise processes advances research in the optimisation and operations research area. In this work, we contribute to both research areas by combining the benefits of both of them. We aim at a dynamic and automated optimisation that can reason on the current situation in which the system is situated and—using this information—exchange the adaptation planning algorithm as well as its parametrisation. Therefore, we propose a framework comprised of several layers that contain the application, an adaptation planning layer, and a meta-optimisation layer where a situation-awareness, an algorithm selection, and a learning component are located. The approach also contains a control loop based on the LRA-M loop known from the Self-aware Computing research. We discuss the applicability of the framework on use cases from Industry 4.0 and Intelligent Transportation Systems. For example, we present the results of a first feasibility study in the domain of Industry 4.0 by minimising setup times in a production environment with multiple machines. Further, we show the proposed framework's meaningfulness for production logistics and storage assignment and order picking scenarios. In the domain of Intelligent Transportation Systems, we analyse the use cases of vehicle routing and platooning coordination. Finally, we derive a set of research questions that will guide the research on this framework in the future.

References

1. Bergenhem, C., Shladover, S., Coelingh, E., Englund, C., Tsugawa, S.: Overview of platooning systems. In: Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012) (2012)
2. Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J., Lang, M.: mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions. arXiv preprint arXiv:1703.03373 (2017)
3. Braekers, K., Ramaekers, K., Van Nieuwenhuysse, I.: The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering* 99, 300–313 (2016)
4. Chai, J., Chang, J., Zhao, Y., Liu, H.: An auto-ml framework based on gbdt for lifelong learning. arXiv preprint arXiv:1908.11033 (2019)
5. Fredericks, E.M., Gerostathopoulos, I., Krupitzer, C., Vogel, T.: Planning as Optimization: Dynamically Discovering Optimal Configurations for Runtime Situations. In: Proc. SASO (2019)
6. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research* 203(3), 539–549 (2010)
7. Hromkovic, J.: *Theoretische Informatik. Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kryptographie*, (2002)
8. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* 36(1), 41–50 (2003)
9. Kinneer, C., Coker, Z., Wang, J., Garlan, D., Goues, C.L.: Managing Uncertainty in Self-Adaptive Systems with Plan Reuse and Stochastic Search. In: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems. pp. 40–50 (2018)
10. Kounev, S., Lewis, P., Bellman, K.L., Bencomo, N., Camara, J., Diaconescu, A., Esterle, L., Geihs, K., Giese, H., Götz, S., et al.: The notion of self-aware computing. In: *Self-Aware Computing Systems*, pp. 3–16. Springer (2017)
11. Krupitzer, C., Roth, F.M., VanSyckel, S., Schiele, G., Becker, C.: A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17, 184–206 (2015)
12. Lesch, V., Krupitzer, C., Stubenrauch, K., Keil, N., Becker, C., Kounev, S., Segata, M.: A comparison of mechanisms for compensating negative impacts of system integration. *Future Generation Computer Systems* 116, 117–131 (March 2020)
13. Lesch, V., Krupitzer, C., Tomforde, S.: Multi-objective Optimisation in Hybrid Collaborating Adaptive Systems. In: Proceedings of the 7th edition in the Series on Autonomously Learning and Optimising Systems (SAOS), co-located with 32nd GI/ITG ARCS 2019. Gesellschaft fuer Informatik (GI) (May 2019)
14. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 18(1), 6765–6816 (2017)
15. Liebetruht, T., Merkl, L.: *Routenzugplanung*. Springer (2018)
16. Moreno, G.A., Papadopoulos, A.V., Angelopoulos, K., Cámara, J., Schmerl, B.: Comparing Model-Based Predictive Approaches to Self-Adaptation: CobRA and PLA. In: 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). pp. 42–53. IEEE (2017)
17. Müller-Schloer, C., Tomforde, S.: *Organic Computing-Technical Systems for Survival in the Real World*. Springer (2017)

18. Nyhuis, P., Wiendahl, H.P.: *Fundamentals of production logistics: theory, tools and applications*. Springer Science & Business Media (2008)
19. Perrouin, G., Morin, B., Chauvel, F., Fleurey, F., Klein, J., Le Traon, Y., Barais, O., Jézéquel, J.M.: *Towards Flexible Evolution of Dynamically Adaptive Systems*. In: 2012 34th International Conference on Software Engineering (ICSE). pp. 1353–1356. IEEE (2012)
20. Porter, B., Rodrigues Filho, R.: *Losing Control: The Case for Emergent Software Systems using autonomous Assembly, Perception, and Learning*. In: 2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO). pp. 40–49. IEEE (2016)
21. Prothmann, H., Tomforde, S., Lyda, J., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: *Self-organised routing for road networks*. In: *Self-Organizing Systems - 6th IFIP TC 6 International Workshop, IWSOS 2012, Delft, The Netherlands, March 15-16, 2012*. Proceedings. pp. 48–59 (2012)
22. Rao, S.S.: *Engineering Optimization: Theory and Practice*. John Wiley & Sons (2009)
23. Sun, X., Lin, J., Bischl, B.: *Reinbo: Machine learning pipeline search and configuration with bayesian optimization embedded reinforcement learning*. arXiv preprint arXiv:1904.05381 (2019)
24. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: *Auto-weka: Combined selection and hyperparameter optimization of classification algorithms*. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 847–855 (2013)
25. Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., Schmeck, H.: *Observation and control of organic systems*. In: *Organic Computing—A Paradigm Shift for Complex Systems*, pp. 325–338. Springer (2011)
26. Van Gils, T., Ramaekers, K., Caris, A., de Koster, R.B.: *Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review*. *European Journal of Operational Research* 267(1), 1–15 (2018)
27. van Willigen, W., Haasdijk, E., Kester, L.: *A multi-objective approach to evolving platooning strategies in intelligent transportation systems*. In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. pp. 1397–1404 (2013)
28. Zhang, J., Wang, F.Y., Wang, K., Lin, W.H., Xu, X., Chen, C.: *Data-driven intelligent transportation systems: A survey*. *IEEE Transactions on Intelligent Transportation Systems* 12(4), 1624–1639 (2011)

Research Challenges in Adaptive Production Systems

Martin Neumayer

Institute for Software & Systems Engineering, University of Augsburg, Germany
neumayer@isse.de

Abstract. In times of personalised products, fluctuating demands and ever-increasing complexity in hard- and software, production systems crave for flexibility and robustness. Self-organisation can help to achieve these goals as self-organising systems autonomously monitor themselves and their environment and adapt to changes observed. Despite extensive study, researchers have hardly addressed some aspects of self-organising production systems. Therefore, we identify three areas to contribute to the vision of self-organising production systems: We plan to extend product descriptions to be more realistic. We further intend to investigate extensions to dynamic scheduling in self-organising production systems. Lastly, we present an approach to avoid deadlocks in self-organising production systems that handle multiple types of products at once.

Keywords: Self-organisation, Production systems, Manufacturing systems, Autonomous systems.

3.1 Introduction

This section introduces the paradigms of organic computing and adaptive systems. It further motivates the application of these paradigms to the manufacturing domain. Lastly, it covers previous work on a special class of adaptive systems, so-called product-flow systems to conclude with open research questions that have hardly been discussed in previous work.

3.1.1 Organic Computing and Adaptive Systems

Organic Computing [40] is an initiative that aims to develop technical systems that exhibit life-like properties, often found in biological systems. Most prominently these life-like properties include robustness and flexibility against disturbances [39]. To achieve these properties, Organic Computing systems observe their environment and adapt autonomously to changes observed by manipulating their environment accordingly. This involves a paradigm shift: Instead of human engineers taking decisions at design time, we are now facing adaptive systems deciding at runtime [23].

To fulfil these requirements, Organic Computing systems are designed to feature self-* properties, including self-configuration and self-organisation. Self-configuration is the ability of a system to change its parameters according to user goals. Self-organisation describes systems autonomously changing their structure to accomplish higher-level goals [40].

3.1.2 Motivation

The vision of applying the paradigm of adaptive systems to manufacturing is long-standing, with publications dating back to the nineties [24]. Since then, the topic has gained additional traction, as the manufacturing domain experiences a shift from mass production to producing customised and even individual products. This shift is accompanied by volatile markets and fluctuating demand. At the same time, production systems consist of many increasingly complex and interconnected hard- and software components. To adapt to these new circumstances, manufacturers focus on gaining flexibility and robustness instead of solely increasing throughput. Adaptive manufacturing systems offer a way to gain these properties:

1. **Robustness:** Adaptive production systems can deal with partial breakdowns by detecting faults and finding new paths of production at runtime.
2. **Flexibility:** Adaptive production systems offer flexibility in terms of the products manufactured and their quantity. As long as the needed capabilities for a new product exist in the system, agents in the system can find new paths applying the methods mentioned above. Adaptive production systems also enable flexibility in terms of the objectives pursued, such as high throughput or low energy consumption.

3.1.3 Background

One way of reaching robustness and flexibility for a special class of systems has been explored in previous work [14, 26, 35], so-called *resource-flow systems* or *product-flow systems*¹. Product-flow systems contain *agents* dispatching, transporting, processing or collecting products. *Storages* are agents dispatching and collecting products. Agents, transporting products from one processing agent to another, are referred to as *autonomous guided vehicles* (AGVs). *Processing agents* may offer several *capabilities* to process a product, such as drilling. A *task*, the blueprint on how to manufacture a product, is described as a sequence of capabilities. Matching the capabilities and transports needed to manufacture a product and the capabilities offered by the agents is termed *reconfiguration*. Reconfiguration can be seen as a form of task allocation [6] or as a scheduling subproblem. The problem of reconfiguration is formulated as a Constraint Satisfaction Problem (CSP) [3]. This CSP can then be solved at runtime in different ways, e.g., centrally using a constraint solver [25] or through coalition

¹ In contrast to previous work, we prefer the notion of product-flow systems as the word resource is ambiguous in the manufacturing domain: It can serve as a term for a machine as well as for a product [6].

formation [26]. We denote the result of reconfiguration, i.e., the product's path through production, as *product flow*.

3.1.4 Research Questions

Despite these promising characteristics and ongoing research, some issues have hardly been discussed in previous work. Therefore, we plan to contribute to the vision of adaptive and self-organising production systems, especially product-flow systems. Our research is guided by three partially interconnected questions.

How to ensure wide applicability? Previous work [26, 35] shows that adaptive production systems can be implemented. However, there are still limitations, e.g., avoiding deadlocks while supporting multiple types of products at a time [37] or allowing task descriptions beyond ordered sequences of capabilities [28]. We plan to extend previous work to overcome these limitations and therefore ensure applicability.

How to ensure performance and scalability? Overcoming limitations such as deadlocks and simplified task descriptions might increase complexity. Thus, we have to re-evaluate the methods used, also considering the scalability necessary for practical application. Concrete research questions subsumed by this main question are whether the constraint-based approach is still suitable and whether decentralisation in the sense of distributed constraint optimisation can increase performance and scalability.

How to achieve openness for human intervention? Lastly, being open for human intervention is one integral feature of organic computing systems [33, 36, 40]. However, this aspect has hardly been studied in the context of adaptive production systems. Therefore, we want to investigate the role of humans in adaptive production systems: How can humans intervene and pose new constraints to adaptive production systems? Is operation according to user-given constraints opposed to performance? Or can human expertise help to relax problems?

From these research questions we derive three research challenges in Section 3.2: Section 3.2.1 discusses the shortcomings of modelling tasks as a sequence of capabilities and presents our planned contributions to address the problem. We cover approaches towards the problem of dynamic scheduling in Section 3.2.2. In Section 3.2.3, we examine the problem of deadlocks in self-organising production systems and briefly summarise an approach to avoid deadlocks in adaptive production systems manufacturing multiple types of products at once. Section 3.3 concludes this paper.

3.2 Research Challenges

This section presents the identified challenges in greater detail. The description of the individual problems adheres to the following structure: Related work, our (planned) contribution and plans to evaluate our contribution.

3.2.1 Realistic Descriptions of Task and Capabilities

3.2.1.1 Related Work

Several publications describe a task as an ordered sequence of capabilities that are executed one after another, altering one particular product [26, 41, 45]. This modelling of a task contradicts practice in many areas of application [28]. E.g., in the furniture industry wooden panels are sawn into several workpieces that are machined individually and later assembled to make up the final product [28, 38].

Keddis et al. [17] refer to splitting an intermediate product or raw material into several as a fork task. A fork task also splits the production process into two parallel processes. In contrast, a synchronisation step synchronises two or more parallel processes. Furthermore, there are cases where capabilities can be replaced by other capabilities (selective tasks) or executed in arbitrary order [17]. Qiao et al. describe similar structures in [32]. Figure 3.1 visualises the different task structures mentioned.

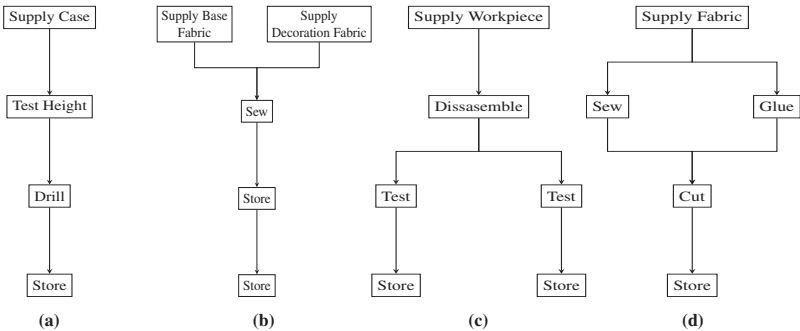


Fig. 3.1. Visualisation of different task structures according to [17]: (a) Sequential task, (b) synchronisation task, (c) fork task, (d) selective task.

The modelling of capabilities has to be more realistic as well. E.g., stating that an agent can perform the capability ‘drill’ does not satisfy the need for practical application. A realistic capability description encompasses parameters describing the material, geometry, and process [17]. A description of the materials used is needed to determine whether an agent can perform the required capability: An agent might be able to drill a piece of wood, while the same agent might not be able to drill a piece of metal. Alike, a description of the product’s geometry is needed to check whether an agent can handle and execute a capability on a product. Due to specific grippers or fixtures, geometry may prevent the execution of a capability. Lastly, process-related information is needed. In our drilling example, we might need to know the exact position, depth, and diameter of the hole. Process-related information should also contain auxiliary materials, such as screws if needed. Depending on the process, related information can take different forms. Therefore, flexible data structures are required.

The increase of capabilities combined with the variety of data needed to describe a task might also turn the modelling of tasks into a tedious and error-prone duty. Here another challenge arises: Generating valid task descriptions from user input [31] such as 3D models. E.g., Lau et al. demonstrate how 3D models of furniture can automatically be split into parts and connectors, using formal grammars [19].

3.2.1.2 Contribution

Based on the related work presented, we identify the following areas of contribution:

1. Survey of descriptions: While the authors in [17] present a solution for the realistic description of task and capabilities, they also state that there might be other methods, e.g., the Business Process Model and Notation (BPMN). A survey will help to compare different approaches and identify the advantages and disadvantages of the approaches.
2. Implementation: After comparing different approaches, we will implement one or several promising approaches for realistic task and capability description. The implementation should include a user interface to create task descriptions, as well as suitable data structures. Using a graph-based structure seems promising.
3. Finding suitable approaches to task allocation and product routing: Differentiating between capabilities with different parameters will lead to an increase in overall capabilities. Together with a realistic description of tasks, the problem of task allocation might turn out more complex. We will have to re-evaluate the constraint-based approach and compare it to other approaches to clarify, which approaches are best suited to these requirements.
4. Generating task descriptions from user input: The approach of Lau et al. [19] seems like a first step in this direction. We plan to reimplement and extend the approach to generate a task description from the parts and connectors.

3.2.1.3 Evaluation

We plan to evaluate the contributions on a showcase basis, i.e., we will provide some showcase products, possibly from the furniture domain, and check if the implemented task and capability description can capture these products. Further, we can compare different methods of task allocation with the provided descriptions, e.g., in terms of runtime. Finally, using a 3D model of the showcase product, we can verify that a valid task description can be generated automatically.

3.2.2 Dynamic Scheduling

3.2.2.1 Related Work - Traditional Scheduling Approaches

Controlling production facilities is a well-studied subject. Researchers have been studying job shop scheduling problems (JSSP) as an NP-hard combinatorial optimisation problem since the 1950s. In a job shop, there is a finite set of products or jobs to

be processed on a finite set of machines. Every product might have a different task, comprised of a set of capabilities. These capabilities must be performed in the given order. Every machine is specialised for its operation, i.e., it offers only one capability. Also, machines can only process one product at a time without the possibility of preemption [5].

With the advent of flexible and reconfigurable manufacturing systems, where machines can perform different capabilities [18], the focus of research has extended to the flexible job shop scheduling problem (FJSSP). Here a machine may offer several capabilities, but switching between capabilities requires a setup time. Thus, an FJSSP can be divided into two subproblems [5, 41]:

1. Assignment of operations to suitable machines.
2. Sequencing of operations on all selected machines to obtain a schedule.

We additionally focus on the subproblem of routing and transporting the products to the machines selected in the assignment. Research on the classical FJSSP often neglects this aspect [5]. The notion of job shop scheduling problems with transportation resources [29] extends the JSSP by a set of identical vehicles that can transport any product. Whenever a product changes from one machine to another, a vehicle must be scheduled to do the transport. Transportation times depend on the machines involved [29].

The goal of all problem variants is to produce a feasible schedule that includes all products or jobs. Furthermore, this schedule should minimise (or maximise) one or several predefined objectives, such as the overall makespan, tardiness, lateness or machine workload, considering transportation and setup times [5]. Recently, objectives considering the environmental impact, e.g., energy consumption, are becoming increasingly relevant in scheduling [22].

Chaudhry and Khan reviewed the techniques used to solve FJSSP problems in [5] to conclude that most of the studied journal contributions devised hybrid techniques (35%) or some form of evolutionary algorithm (24%), e.g., genetic algorithms, differential evolution or learning classifier systems. The authors define hybrid techniques as techniques that combine one or several (meta-) heuristics to benefit from their strengths [5]. About 10% of the authors used deterministic heuristics, while tabu search was used in 6% of the cited papers. Other techniques include integer/linear programming and mathematical programming, as well as nature-inspired algorithms such as particle swarm optimisation, simulated annealing, ant colony optimisation, or artificial bee colony [5]. Scott et al. investigated whether human expertise can help to solve hard optimisation problems such as routing or scheduling [34]. In human-computer optimisation, humans and computers collaborate, e.g., a user specifies a search space that the computer then explores. Scott et al. conclude that human expertise can indeed help to manage the usage of computational resources in optimisation [34].

Due to the complexity, researchers often tackle JSSP variants by splitting the problem into the aforementioned subproblems and solving them one after another [45]. Researchers also assume a deterministic environment [4] and omit complex constraints, e.g., regarding uncertain processing, transportation or setup times, maintenance, or machine breakdown to facilitate the problems [5]. Another method to

relax the problem is to decrease the time horizon of the schedule [45]. However, beyond these simplifications, manufacturing systems are characterised by unpredictable events and disturbances [21, 30, 45]. Therefore, authors doubt whether centralised approaches can cope with the dynamic and sometimes even chaotic nature of production systems [6, 45] and provide the required flexibility [21].

3.2.2.2 Related Work - Dynamic Scheduling through Self-Organisation

Instead of computing a schedule upfront using global knowledge, research in adaptive production systems has focused on solving the problems of assignment, sequencing, and routing through the interaction of the involved agents. This leads to a different focus: From finding an optimal to finding a dynamic schedule [30, 42]. In return, researchers hope to achieve greater robustness, flexibility and scalability.

Different authors [13, 20, 42, 45] have devised potential field approaches to solve the assignment and routing subproblems and guide products through production: On the one hand, machines send out potential fields to attract empty vehicles or vehicles carrying products. On the other hand, vehicles sense the attraction fields sent out by machines, decide for one and move towards it. Figure 3.2 shows the local interaction between vehicles or AGVs and machines in [13].

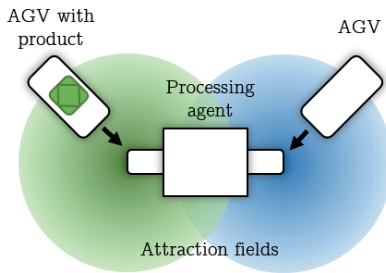


Fig. 3.2. Local interaction between a processing agent and AGVs, adapted from [13]: The input buffer on the left sends out a potential field to attract AGVs carrying products, while the output buffer on the right emits a potential field to attract empty AGVs that remove products from the output buffer.

While the approaches are conceptually similar, they differ in many details: Attraction fields can encode a simple enumeration of product types [13] or include more complex concepts and constraints such as product size, quality of service, availability and workload of machines [42, 45]. Routing can take place on a fixed graph that represents routes of a shuttle system [20, 45] or a general two-dimensional space [13, 42]. Lastly, the control of attraction fields can be hardcoded [42] or learned, e.g., by reinforcement learning [13].

The potential field approach exhibits strong self-organisation, as it requires no central control [10]. However, quantitative analysis is challenging due to its dynamic

nature [42]. Therefore, researchers resort to an experiment-based analysis: They measure objectives during simulation [13] or produce a schedule by running a simulation [45]. This schedule or data is afterwards analysed in terms of optimality. The experiment-based analysis does not allow for behavioural guarantees, which are indispensable for production systems.

The Restore Invariant Approach (RIA) [26] tries to fill this gap by specifying and enforcing a corridor of correct behaviour. Correct behaviour includes a feasible assignment of machines and correct routing of products. Sequencing of products is not part of the behavioural corridor. Instead, it arises as an emergent property. The agents monitor the corridor to ensure that the agent that detects a violation starts a reconfiguration. The purpose of reconfiguration is bringing the system back into the corridor. Reconfiguration can be centralised [25] or partly decentralised using coalition formation [35]. In the centralised variant, a central controller collects information about all agents and can then solve the problems of assignment and routing by applying constraint solving or a genetic algorithm. In the decentral reconfiguration, the agent noticing the violation (leader) forms a coalition with its neighbouring agents. The leader then tries to solve the problems of assignment and routing using the information from its neighbours. If the leader can't solve the problem, he enlarges the coalition and re-tries to solve the problem until he finds a solution [26]. A verified result checker then reviews the found solution before the leader distributes it among the agents in the coalition. The verified result checker together with the verification of the functional system allows guaranteeing that the system behaves as intended [26,27].

3.2.2.3 Contribution

Building upon previous and related work, there are several areas of contribution:

1. The first area of contribution is related to the realistic description of tasks and capabilities presented in Section 3.2.1. We plan to investigate how these realistic descriptions affect finding a solution towards the assignment and product routing in the context of the RIA. We assume that the realistic descriptions will increase the complexity of the problems. Thus, we plan to re-evaluate the use of constraint solving and genetic algorithms in comparison to other optimisation or learning methods.
2. The second area of contribution is concerned with comparing the mechanisms presented before: Can the different mechanisms profit from another? E.g., can we get rid of partly centralised control of the coalition leader in the RIA to achieve strong self-organisation as seen in the potential field approach? As a concrete contribution, we plan to implement and evaluate a reconfiguration mechanism based on distributed constraint optimisation.
3. Third, we plan to examine the use of machine learning techniques for dynamic scheduling. One exemplary use case is predictive maintenance. Researchers already use machine learning algorithms to predict machine or component failure successfully [8]. However, often effective countermeasures besides human intervention are missing. The combination of dynamic scheduling and machine

learning seems promising, as products could be re-assigned and rerouted autonomously in case of imminent failure.

4. Lastly, we plan to investigate the role of humans in dynamic scheduling: Can human expertise help to solve the problem, like in Scott et al.? Can human-computer cooperation help to build understanding and trust in the solution found? We further want to answer the following questions: How can humans intervene and post new constraints? Do those constraints oppose performance and scalability?

3.2.2.4 Evaluation

We plan to evaluate our contribution in comparative studies, where we compare two variants, e.g., with and without realistic task descriptions, in a given scenario. These comparative studies allow us to measure and compare the relevant attributes, e.g., runtime and solution quality. The evaluation should also cover different problem sizes, e.g., number of agents or number of products, to draw conclusions about scalability. Scenarios might also include disturbances, i.e., component or agent failure, to quantify changes in robustness.

3.2.3 Dealing with Deadlocks

Deadlocks are situations where two or more agents are waiting for another to finish in a way that no one ever finishes [9]. The risk of deadlocks in production systems is well-known, and as deadlocks may halt production, they are also heavily studied [1, 16, 37]. Consider the motivating example in Figure 3.3 that demonstrates how a simple cyclic arrangement can lead to a deadlock. Cyclic arrangements concerning multiple tasks are also possible and might be even harder to detect locally.

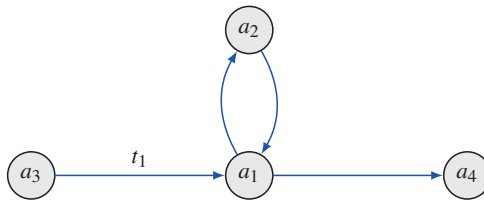


Fig. 3.3. Cyclic arrangement of two agents a_1 and a_2 . The arrows denote the product flow of task t_1 : a_1 receives products from a_3 , processes them and hands them over to a_2 . After processing at a_2 , a_1 receives the products again and applies another capability before handing them over to a_4 . If a_1 accepts a product from a_3 while a_2 also holds a product a deadlock emerges.

3.2.3.1 Related Work

As Figure 3.3 suggests, deadlocks in manufacturing systems are caused by cycles. Specifically, Wysk et al. proof that the following two conditions must be met for a deadlock to occur [7, 44]:

1. There has to be at least one cycle in the product flow.
2. Each agent in the cycle has to be occupied by a product.

To deal with deadlocks, researchers devised a variety of methods, including Petri nets [1, 43] that restrict the agent's actions to prevent deadlocks. Event-based approaches [11, 12] use global knowledge to detect cycles and decide on save transactions. However, as both methods require global knowledge or control, they are not suitable for distributed systems.

Distributed cycle detection algorithms, such as the one presented in [2], detect cycles by passing messages between the agents. Messages are forwarded until they return to their sender or they reach the end of the system and cannot be forwarded any further. This algorithm allows determining whether an agent is in a cycle. Though, it does not provide additional information, such as the cycle's size, which is essential for avoiding deadlocks in a distributed manner.

Lastly, we directly build upon the work of Steghöfer et al. [37]. In their work, the authors present a decentralised deadlock avoidance approach based on message passing. However, dealing with multiple types of products is left as future work.

3.2.3.2 Contribution

Thus, in [15], we present a decentral approach to avoid deadlocks in production systems that handle multiple tasks at once. We refrain from generally averting cycles, as this results in a loss of flexibility. Instead, we rely on the aforementioned theoretical insight of Wysk et al. To prevent that each agent is occupied by a product, we employ a two-step procedure [15]:

1. Cycle detection: Whenever the configuration of the system changes, e.g., due to a new type of product or the (partial) failure of an agent, agents send out messages to detect cyclic arrangements. Cycles are then stored alongside the number of products that are allowed to enter.
2. Enforcing the limits for products in cycles: When production resumes, agents keep track of the number of products that are currently in each cycle. The agents that are entrances and exits of the cycles enforce the limits calculated in cycle detection by coordinating through message-passing.

3.2.3.3 Evaluation

To evaluate our approach experimentally, we run several simulations with different configurations and measure the number of deadlocks encountered, the runtime needed, and the number of messages sent. Additionally, we calculate the system's throughput

by dividing the number of manufactured products by the runtime. Our results suggest that our approach effectively avoids deadlocks in the configurations considered. Furthermore, our approach outperforms a simple conservative locking algorithm in terms of message overhead and runtime. Therefore, systems using our approach can realise higher throughput compared to systems using the conservative locking algorithm [15].

3.2.3.4 Future Work

Despite the encouraging results, some challenges remain: First, our experimental evaluation does not formally prove the deadlock avoiding property of our algorithm. Therefore, we strive for formal proof confirming our experimental results. Additionally, the experimental configurations only cover a small set of agents. To ensure the scalability of our approach, we plan to conduct experiments with a larger number of agents and also investigate the message overhead in a formal way. Lastly, we plan to examine whether adding soft constraints that favour solutions without cycles to our constraint model can relax the problem.

3.3 Conclusion

In this paper, we summarise research questions in adaptive production systems. Namely, we suggest using more realistic task descriptions, including structures such as selective tasks, forks, and synchronisations. Further, we plan to extend capability descriptions to contain material-, geometry-, and process-related information. The effects of elaborating task and capability descriptions on the problem of task allocation have to be studied. We further plan to direct research to automatically generating task descriptions from user input, as manually creating task descriptions becomes more complex and error-prone.

In times of fluctuating markets, dynamic control is another key-issue for adaptive production systems. We present different approaches towards the problem of dynamic scheduling and propose to take advantage of the combination of the different concepts. We intend to allow realistic task structures and human intervention in dynamic scheduling. We further plan to integrate machine learning techniques into adaptive production systems to benefit from the rapid progress in this area. Combining machine learning and self-organisation allows to detect failures beforehand and offer countermeasures such as rerouting products. Therefore, the combination may further increase the robustness of adaptive production systems.

Another problem in flexibly linked, decentral production systems with multiple tasks is dealing with deadlocks. To handle both, the decentral nature of adaptive systems as well as many products at a time, we present a message-based deadlock avoidance approach in [15]. Our experimental evaluation shows that the approach avoids deadlocks in several realistic system configurations with reasonable message overhead. However, evaluating the scalability of our approach in larger configurations, as well as formally proving the deadlock-avoiding property remains as future work.

References


1. Banaszak, Z.A., Krogh, B.H.: Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Transactions on robotics and automation* 6(6), 724–734 (1990)
2. Boukerche, A., Tropper, C.: A distributed graph algorithm for the detection of local cycles and knots. *IEEE Transactions on Parallel and Distributed Systems* 9(8), 748–757 (1998)
3. Brailsford, S.C., Potts, C.N., Smith, B.M.: Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research* 119(3), 557 – 581 (1999)
4. Chaari, T., Chaabane, S., Aissani, N., Trentesaux, D.: Scheduling under uncertainty: Survey and research directions. In: 2014 International Conference on Advanced Logistics and Transport (ICALT). pp. 229–234 (2014)
5. Chaudhry, I.A., Khan, A.A.: A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research* 23(3), 551–591 (2016)
6. Chevalyre, Y., Endriss, U., Lang, J., Dunne, P., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-Aguilar, J., Sousa, P.: Issues in multiagent resource allocation. *Informatica* 30, 3–31 (2006)
7. Cho, H., Kumaran, T., Wysk, R.A.: Graph-theoretic deadlock detection and resolution for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation* 11(3), 413–421 (1995)
8. Cline, B., Niculescu, R.S., Huffman, D., Deckel, B.: Predictive maintenance applications for machine learning. In: 2017 Annual Reliability and Maintainability Symposium (RAMS). pp. 1–7 (2017)
9. Coffman, E.G., Elphick, M., Shoshani, A.: System deadlocks. *ACM Computing Surveys (CSUR)* 3(2), 67–78 (1971)
10. Di Marzo Serugendo, G., Gleizes, M.P., Karageorgos, A.: Self-organization in multi-agent systems. *Knowledge Engineering Review* 20(2), 165–189 (2005)
11. Fanti, M.P., Maione, B., Mascolo, S., Turchiano, A.: Event-based feedback control for deadlock avoidance in flexible production systems. *IEEE Transactions on Robotics and Automation* 13(3), 347–363 (1997)
12. Fanti, M.P., Zhou, M.: Deadlock control methods in automated manufacturing systems. *IEEE Transactions on systems, man, and cybernetics-part A: systems and humans* 34(1), 5–22 (2004)
13. Fujii, N., Hatono, I., Ueda, K.: Reinforcement learning approach to self-organization in a biological manufacturing system framework. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 218(6), 667–673 (2004)
14. Güdemann, M., Ortmeier, F., Reif, W.: Formal modeling and verification of systems with self-x properties. In: Yang, L.T., Jin, H., Ma, J., Ungerer, T. (eds.) *Autonomic and Trusted Computing*. pp. 38–47. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
15. Hirsch, J., Neumayer, M., Ponsar, H., Kosak, O., Reif, W.: Deadlock avoidance for multiple tasks in a self-organizing production cell. In: 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS). pp. 178–187 (2020)
16. Hu, H., Li, Z.: Local and global deadlock prevention policies for resource allocation systems using partially generated reachability graphs. *Computers & Industrial Engineering* 57(4), 1168 – 1181 (2009)
17. Keddis, N., Kainz, G., Zoitl, A., Knoll, A.: Modeling production workflows in a mass customization era. In: 2015 IEEE International Conference on Industrial Technology (ICIT). pp. 1901–1906. IEEE (2015)
18. Koren, Y.: *The global manufacturing revolution: product-process-business integration and reconfigurable systems*. John Wiley & Sons (2010)

19. Lau, M., Ohgawara, A., Mitani, J., Igarashi, T.: Converting 3d furniture models to fabricatable parts and connectors. *ACM Trans. Graph.* 30(4) (Jul 2011)
20. Leitão, P., Barbosa, J., Trentesaux, D.: Bio-inspired multi-agent systems for reconfigurable manufacturing systems. *Engineering Applications of Artificial Intelligence* 25(5), 934–944 (2012)
21. Leitão, P.: Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence* 22(7), 979 – 991 (2009), distributed Control of Production Systems
22. May, G., Stahl, B., Taisch, M., Prabhu, V.: Multi-objective genetic algorithm for energy-efficient job shop scheduling. *International Journal of Production Research* 53(23), 7071–7089 (2015)
23. Müller-Schloer, C., Tomforde, S.: *Organic Computing - Technical Systems for Survival in the Real World*. Birkhäuser (2017), <https://doi.org/10.1007/978-3-319-68477-2>
24. N. Kubota, T. Fukuda, F. Arai, K. Shimojima (eds.): Genetic algorithm with age structure and its application to self-organizing manufacturing system: ETFA '94. 1994 IEEE Symposium on Emerging Technologies and Factory Automation. (SEIKEN) Symposium) -Novel Disciplines for the Next Century- Proceedings (1994)
25. Nafz, F., Ortmeier, F., Seebach, H., Steghöfer, J.P., Reif, W.: A universal self-organization mechanism for role-based organic computing systems. In: González Nieto, J., Reif, W., Wang, G., Indulska, J. (eds.) *Autonomic and Trusted Computing*. pp. 17–31. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
26. Nafz, F., Seebach, H., Steghöfer, J.P., Anders, G., Reif, W.: *Constraining Self-organisation Through Corridors of Correct Behaviour: The Restore Invariant Approach*, pp. 79–93. Springer Basel, Basel (2011)
27. Nafz, F., Seebach, H., Steghöfer, J.P., Bäuml, S., Reif, W.: A formal framework for compositional verification of organic computing systems. In: Xie, B., Branke, J., Sadjadi, S.M., Zhang, D., Zhou, X. (eds.) *Autonomic and Trusted Computing*. pp. 17–31. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
28. Neumayer, M.: Towards realistic task and capability descriptions in self-organizing production systems. In: *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. pp. 234–236 (2020)
29. Nouri, H.E., Driss, O.B., Ghédira, K.: A classification schema for the job shop scheduling problem with transportation resources: State-of-the-art review. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. (eds.) *Artificial Intelligence Perspectives in Intelligent Systems*. pp. 1–11. Springer International Publishing, Cham (2016)
30. Ouelhadj, D., Petrovic, S.: A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling* 12(4), 417 (2009)
31. Pfrommer, J., Schleipen, M., Beyerer, J.: Pprs: Production skills and their relation to product, process, and resource. In: *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*. pp. 1–4 (2013)
32. Qiao, L., Kao, S., Zhang, Y.: Manufacturing process modelling using process specification language. *The International Journal of Advanced Manufacturing Technology* 55(5-8), 549–563 (2011)
33. Schmeck, H.: Organic computing - a new vision for distributed embedded systems. In: *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*. pp. 201–203 (2005)
34. Scott, S.D., Lesh, N., Klau, G.W.: Investigating human-computer optimization. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. pp. 155–162 (2002)

35. Seebach, H., Nafz, F., Steghöfer, J.P., Reif, W.: How to Design and Implement Self-organising Resource-Flow Systems, pp. 145–161. Springer Basel, Basel (2011)
36. Steghöfer, J.P., Kiefhaber, R., Leichtenstern, K., Bernard, Y., Klejnowski, L., Reif, W., Ungerer, T., André, E., Hähner, J., Müller-Schloer, C.: Trustworthy organic computing systems: Challenges and perspectives. In: Xie, B., Branke, J., Sadjadi, S.M., Zhang, D., Zhou, X. (eds.) *Autonomic and Trusted Computing*. pp. 62–76. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
37. Steghöfer, J.P., Mandrekar, P., Nafz, F., Seebach, H., Reif, W.: On deadlocks and fairness in self-organizing resource-flow systems. In: Müller-Schloer, C., Karl, W., Yehia, S. (eds.) *Architecture of Computing Systems - ARCS 2010*. pp. 87–100. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
38. Tippayawong, K.Y., Prapasirisulee, T.: Productivity enhancement in a wood furniture manufacturing factory by improving work procedures and plant layout. *Recent Advances in Manufacturing Engineering* pp. 30–34 (2011)
39. Tomforde, S., Kantert, J., Müller-Schloer, C., Bödel, S., Sick, B.: Comparing the effects of disturbances in self-adaptive systems - A generalised approach for the quantification of robustness. *Trans. Comput. Collect. Intell.* 28, 193–220 (2018)
40. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic computing in the spotlight. *arXiv preprint arXiv:1701.08125* (2017)
41. Trentesaux, D., Pach, C., Bekrar, A., Sallez, Y., Berger, T., Bonte, T., Leitão, P., Barbosa, J.: Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice* 21(9), 1204–1225 (2013)
42. Vaario, J., Ueda, K.: An emergent modelling method for dynamic scheduling. *Journal of Intelligent Manufacturing* 9(2), 129–140 (1998)
43. Wu, N., Zhou, M.: Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35(6), 1193–1202 (2005)
44. Wysk, R.A., Yang, N.S., Joshi, S.: Detection of deadlocks in flexible manufacturing cells. *IEEE Transactions on robotics and automation* 7(6), 853–859 (1991)
45. Zbib, N., Pach, C., Sallez, Y., Trentesaux, D.: Heterarchical production control in manufacturing systems using the potential fields concept. *Journal of Intelligent Manufacturing* 23(5), 1649–1670 (2012)

Self-Learning Governance of Competitive Multi-Agent Systems

Michael Pernpeintner

OrcidID 0000-0001-6939-1028
Institute for Enterprise Systems (InES),
University of Mannheim, Germany
pernpentner@es.uni-mannheim.de

Abstract. Multi-Agent Systems (MAS) are widely used as a succinct model for distributed systems with (partly or fully) autonomous components. Whenever these components do not intrinsically cooperate, but pursue their individual goals in a purely selfish way (*Competitive MAS*), there is a natural challenge to prevent undesirable and destructive system behaviour and to achieve system-level objectives.

While agent autonomy is an essential characteristic of an MAS and can therefore not simply be replaced with full control or centralised management without losing its core functionality, it is still possible to achieve a certain level of control by applying a suitable governance approach.

I am proposing a new solution for this challenge. My approach adds to the usual agent/environment structure of an MAS a Governance component which can observe publicly available information about agents and environment, and, in turn, has the right to restrict the action spaces of agents and thus prevent certain environmental transitions.

As opposed to most existing methods, this approach does not rely on any assumptions about agent utilities, strategies or preferences. It therefore takes into consideration the fundamental fact that actions are not always directly linked to genuine agent preferences, but can also reflect anticipated competitor behaviour, be a concession to a superior adversary or simply be intended to mislead other agents.

The present paper motivates and describes the approach, defines the scope of the PhD project and shows its current status and challenges.

Keywords: Multi-Agent System, Competition, Governance, Restriction.

4.1 Introduction

4.1.1 Motivation

An essential feature of Multi-Agent Systems is the fact that agents depend on each other: The way the system behaves is not defined by the actions of one individual agent, but rather by the combination of all actions [18]. Therefore, a single agent can never be certain about the result of a chosen action. This mutual influence leads to

strategic behaviour and sometimes even seemingly erratic actions—especially when an agent is human—, and at the same time decouples *intended* and *observed* system behaviour.

Example 1. Consider an MAS consisting of two agents X and Y , two environmental states A (initial state) and B , and two actions 0 and 1 for each agent, resulting in the joint action set $\{00, 01, 10, 11\}$ (the joint action 10 means that the first agent, X , chooses action 1, while the second agent, Y , takes action 0). The transition function of the MAS is shown in Figure 4.1. Imagine now an observer who sees the following sequence of actions and transitions:

$$A \xrightarrow{10} A \xrightarrow{01} A \xrightarrow{00} B$$

The observer, as is does not know the preferences of X and Y , cannot tell from the observed facts if X wanted to stay at state A and changed its action from 1 in the first step to 0 in the second step because it anticipated Y 's second action, or if X observed the uselessness of its first action and then tried another strategy to reach state B (and failed again). This shows that intentions are not immediately linked to observable behaviour, and, in particular, no preference order over the environmental states can be concluded.

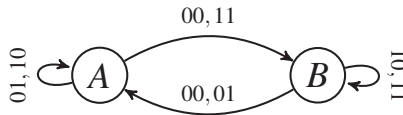


Fig. 4.1. Transition graph of a simple MAS

On the one hand, this is a challenge for a participating agent which needs to derive a strategy to counter its opponents' actions based on what it can see, but on the other hand, it makes it inherently hard to control or steer such a system using an external governing entity. I am specifically interested in the latter case, where there is a system-level objective (or “global desirable properties” [34]) to be achieved in addition to the individual goals of the agents.

It follows that preference elicitation (the process of deriving preferences over states from observed behaviour) is not feasible without additional assumptions about the link between actions and preferences. In general, the resulting preference order might be wrong, and relying on it could therefore lead to false conclusions about target conflicts and controlling decisions.

Nevertheless, the task of governing such an MAS requires some sort of planning and prediction of behaviour: In order to achieve a system-level objective, the Governance needs to prevent transitions which lead to violations of this objective. Therefore, it relies on collecting observable information and deriving knowledge about the future system behaviour. The two fundamental questions that it needs to

answer based on this knowledge are: “*What will agents do next?*” and “*Which actions need to be forbidden in order to prevent undesirable transitions?*”

This online learning mechanism guarantees that the system can self-adapt to both changes in the setup—such as number of agents and system objective—and unforeseen strategic behaviour of the agents. It therefore ensures that the overall MAS is at the same time robust and flexible, without requiring manual intervention at run-time. From an Organic Computing perspective, moving the Governance logic and the selection of restrictions into the system makes the system “organic” in the sense that it can handle human agents in the same way as it handles agents based on experts systems, simple heuristics or sophisticated AI methods, and it therefore serves as a means for flexibly balancing the influences of otherwise uncontrolled agents.

4.1.2 Setting and Contribution

My PhD project is situated in the broader field of Organic Computing [27] and targets the problem of providing governance for competitive Multi-Agent Systems, purely based on the observation of public behaviour, i.e., actions and transitions. In contrast to most existing approaches for Governed MAS or Normative MAS, I argue that it is not reasonable to assume a-priori knowledge of agent utilities, preferences or strategies.

The contribution will consist of a new model for Governed MAS, proof of its feasibility and applicability, thorough analysis with respect to capabilities and complexity, and an evaluation which shows the performance of the framework in a real-world use case. Thereby, the research questions listed in Section 4.3.1 will be answered concisely and in depth.

4.1.3 Structure of the Paper

The remainder of this paper is organised as follows:

Section 4.2 defines the system model and the governing instance. Section 4.3 lists the research questions, shows what has already been accomplished and describes the necessary future work to complete the intended contribution. Section 4.4 recaps relevant existing work and places this project within the context of these approaches, while Section 4.5 outlines a real-world evaluation use case. Finally, Section 4.6 sums the paper up.

A more formal treatment of the multi-attribute case, including a governance algorithm and its evaluation, has recently been submitted [30]. Part of this submission is being included here in shortened form to show motivation, general system model, preliminary results and existing work.

4.2 Model

4.2.1 Agents and Environment

The general MAS model is based on [35]: Consider a finite set $\mathcal{P} = \{p_1, \dots, p_n\}$ of agents (or players). An agent p_i perceives, at every time step $t \in \mathbb{N}_0$, the current state

$s_t \in \mathcal{S}$ of a temporally discretised environment and then acts within this environment by performing an action $a_i \in \mathcal{A}_i$, following a confidential (and not necessarily deterministic) *action policy* $\pi_i: \mathcal{S} \rightarrow \mathcal{A}_i$. The environmental state then changes from time step t to $t+1$ according to the combination of actions (the *joint action* $a = (a_1, \dots, a_n) \in \mathcal{A}$) taken by the agents, as expressed by a *transition function* $\delta: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$.

Definition 1. A Multi-Agent System is the 6-tuple

$$\mathcal{M} = (\mathcal{P}, \mathcal{S}, \mathcal{A}, \pi, \delta, s_0) .$$

4.2.2 Governance

In the basic MAS model of Section 4.2.1, the evolution of an MAS from t to $t+1$ follows the formula

$$s_{t+1} = \delta(s_t, \pi(s_t)) .$$

Since the action policies π_i are at the agents' sole discretion, one can see immediately that this progression can be influenced by an external authority via two levers only: Either by changing *what agents can do* (altering their action sets) or by changing *what consequences actions have* (altering the transition function).

The proposed governance model of this paper follows a strict separation of concerns: The transition function represents the unalterable evolution of the environment according to the actions taken by all agents, while the restriction of actions is performed by the Governance and therefore artificial. To use an analogy, the transition function accounts for the laws of nature in the system, whereas the Governance plays the role of the legislature.

4.2.2.1 Observation and Intervention

At the beginning of each cycle t , the Governance defines *allowed actions* before the agents choose their respective actions from this restricted action set:

$$\mathcal{A}_t = \Gamma_{s_{\mathcal{G}}^{(t)}}(s_t) ,$$

where $\mathcal{A}_t \subseteq \bar{\mathcal{A}}$ is a “rectangular” subset of a *fundamental action set* $\bar{\mathcal{A}} = \prod_i \bar{\mathcal{A}}_i$, i.e., $\mathcal{A} = \prod_i \mathcal{A}_i^{(t)}$ with $\mathcal{A}_i^{(t)} \subseteq \bar{\mathcal{A}}_i \forall i$. The subscript in $\Gamma_{s_{\mathcal{G}}^{(t)}}(s_t)$ hints to the fact that Γ implicitly uses as an input not only the current environmental state s_t , but also the internal state $s_{\mathcal{G}}^{(t)} \in \mathcal{S}_{\mathcal{G}}$ of the Governance, which includes the knowledge acquired so far. Since this is always the case, the subscript will henceforth be omitted for brevity. The shape of \mathcal{A}_t needs to be rectangular for the simple reason that agents act independently in each step, which means that it is not possible to make conditional restrictions such as $\mathcal{A}_t = \{(a,x), (b,x), (a,y)\}$ since the Governance cannot, in this example, force p_1 to choose action a whenever p_2 chooses action y .

For each agent p_i , there is a *neutral action* $\emptyset_i \in \bar{\mathcal{A}}_i$ which cannot be deleted from the set of allowed actions. The resulting joint action \emptyset is therefore always allowed.

As soon as all agents have made and communicated their choice of action $a = (a_i)_i \in \mathcal{A}_t$, the Governance can use the information gathered by observing the actions and the subsequent transition to learn about the agents and the effectiveness of Γ . This *learning step* is expressed as an update of the Governance’s internal state which, in turn, will be used by Γ in the next step, i.e.,

$$s_{\mathcal{G}}^{(t+1)} = \lambda \left(s_{\mathcal{G}}^{(t)}, s^{(t)}, a \right).$$

As opposed to some authors [4], I make no distinction between legal and physical power: An agent can choose only from the set of currently allowed actions (which might change from one step to the next), and it is not possible to disobey this rule. Nevertheless, the neutral action ensures that the system can operate with missing or invalid input coming from the agents—it simply uses \emptyset_i as a fallback.

4.2.2.2 System Objective

As mentioned in Section 4.1.1, I assume that there is a certain system objective which is to be fulfilled, in addition to the agent-specific goals (and maybe conflicting with those agent goals). This way, the restriction mechanism of the Governance has the clear purpose of fulfilling this objective. Since the Governance has only probabilistic information about the agents’ future actions, its objective needs to be compatible with probabilistic reasoning and therefore quantifiable.

While the system objective can be an arbitrary function from \mathcal{S} to \mathbb{R} , there are two common types: Either minimising (or maximising) a numerical parameter, which can directly be expressed by $c_{\mathcal{G}}$, or dividing the state space into obeying states \mathcal{S}_+ and violating states $\mathcal{S}_- := \mathcal{S} \setminus \mathcal{S}_+$. In the latter case, the function

$$c_{\mathcal{G}}(s) := \mathbb{1}_{\mathcal{S}_-}(s) \tag{4.1}$$

describes a system objective which prefers all obeying states to all violating states by minimising $c_{\mathcal{G}}$. Therefore, the Governance will pursue an obeying state with minimal restriction of the agents.

Definition 2. *The system objective of an MAS \mathcal{M} is defined as a cost function $c_{\mathcal{G}} : \mathcal{S} \rightarrow \mathbb{R}$ such that the Governance tries to reach and maintain a state of minimal cost.*

This cost function simply defines the preference of the Governance over the states of the environment; it does not necessarily correspond to a “real” cost.

The definition of a *Governed Multi-Agent System* is now that of an MAS, together with a specification of the Governance’s behavior:

Definition 3. *A Governed Multi-Agent System (GMAS) is the 10-tuple*

$$\mathcal{M}_{\mathcal{G}} = \left(\mathcal{P}, \mathcal{S}, \bar{\mathcal{A}}, \pi, \delta, s_0, s_{\mathcal{G}}^{(0)}, c_{\mathcal{G}}, \Gamma, \lambda \right)$$

with $\Gamma : \mathcal{S} \rightarrow 2^{\bar{\mathcal{A}}}$ and $\lambda : \mathcal{S}_{\mathcal{G}} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}_{\mathcal{G}}$.

4.2.3 Run-time Process

The sequence of actions taken by the different components in one time step is shown in Figure 4.2. At each step, the Governance can define allowed actions via Γ (before the agents act) and learn from the observed actions via λ (after the agents have acted). The environment itself is not affected at all by the existence of the Governance.

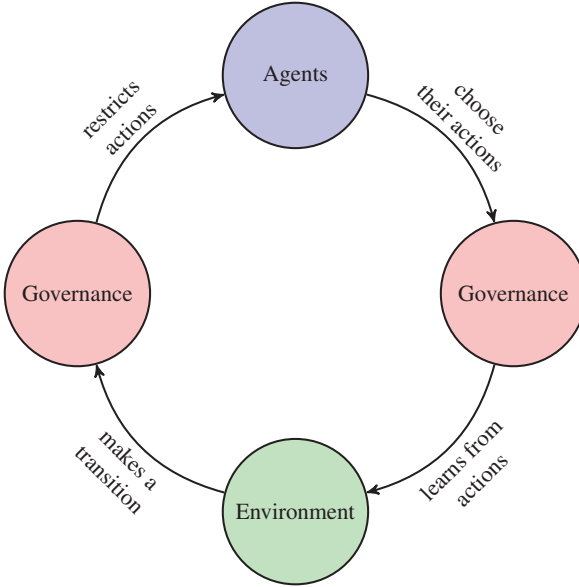


Fig. 4.2. Run-time Process

The performance of the Governance can now be measured by looking at two key parameters: (a) How high is the cost incurred at each state? and (b) How many restrictions were applied to achieve this cost? The second question naturally gives rise to the following notion:

Definition 4. *The degree of restriction of \mathcal{G} at time t is the ratio of forbidden actions and fundamental actions:*

$$\tau_{\mathcal{G}}(t) := 1 - \frac{|\mathcal{A}_t|}{|\bar{\mathcal{A}}|} \in [0, 1]$$

Taking this value as an indicator for Governance performance implies that all actions are equally important. Since this is not always the case, a more elaborate measure (e.g. comparing the size—with respect to some environment-specific metric—of the state set following from taking all actions in \mathcal{A}_t and $\bar{\mathcal{A}}$) might be useful to better capture the “real” magnitude of the Governance-induced restrictions. This is a topic to be examined in future work.

Example 2. Consider a smart home environment consisting of 7 binary variables: $S = T \times O \times W \times B \times H \times L \times A \cong \mathbb{B}^7$, where the variables denote Time (day/night), Occupancy (occupied/empty), Window (open/closed), Blinds (open/closed), Heating (on/off), Lights (on/off) and Alarm (on/off), respectively. n agents, who each have their individual preferences over the state, can now choose to change at most one of the variables W, B, H, L or A (the corresponding actions start at 1) at each step (they cannot, however, influence the Time or the Occupancy of the house). A variable is changed regardless of how many agents have chosen to change it at a single time step.

An exemplary progression of this system could be

$$s_0 = 1100101 \xrightarrow{37\emptyset} 1110100 \xrightarrow{464} 1111110 \\ \xrightarrow{\emptyset\emptyset 5} 1111010 \xrightarrow{564} 1110100 \xrightarrow{436} 1101110,$$

where states are written as binary numbers and there are three agents acting upon the environment with the action sets

$$\bar{\mathcal{A}}_i = \{\emptyset, 3, 4, 5, 6, 7\} \forall i.$$

Time and Occupancy would of course need to be controlled by non-controllable environmental forces, but this is omitted here for simplicity.

Define now a Governance with cost function c_G as in Definition 2 where

$$\mathcal{S}_+ = \{s \in \mathcal{S} : (\bar{w}(s) \vee \bar{h}(s)) \wedge (a(s) \vee o(s)) \wedge (\bar{l}(s) \vee o(s))\},$$

meaning that the system wants to make sure that (a) the window is not open while the heating is turned on, (b) the alarm is on when the house is empty, and (c) the lights are off when there's nobody home. It is therefore the task of the Governance to impose minimal restrictions on the agents while keeping $s_t \in \mathcal{S}_+$.

One can now see that $s_1 = 1110100$ incurs cost $c_G(s_1) = 1$ since $s_1 \notin \mathcal{S}_+$. While the Governance probably cannot anticipate and prevent this transition between $t = 0$ and $t = 1$ due to lack of knowledge, it might be able to do so at a later time when enough information has been gathered. For example, at $t = 3$, the Governance could forbid action $5 \in \mathcal{A}_1$ such that the joint action 564 cannot happen. If p_1 now chooses action 3 instead, $s_4 = \delta(s_3, 364) = 1100000 \in \mathcal{S}_+$, and the Governance has therefore successfully prevented an undesirable transition.

4.3 Scope

4.3.1 Research Questions

The goal of this PhD project is the theoretical foundation, development, analysis and application of a GMAS platform which can be used to govern real-world Multi-Agent Systems with arbitrary agents. Therefore, the following research questions describe the gaps and open challenges in the current state of the art:

- RQ1 Is the observation of actions and transitions, together with hard restriction of action spaces, sufficient and suitable for effective governance with respect to a given system objective? If not, which further assumptions, limitations or relaxations are necessary?
- RQ2 Which data structures and algorithms can be used to create a scalable computation framework which can be used for online (real-time) governance? How does this framework perform in both benchmark and real-world applications?
- RQ3 How can an agent (or a group of agents) manipulate the mechanism, and how can the Governance effectively identify and prevent manipulation?

4.3.2 Current Status

A widely accepted environmental limitation in MAS research is to assume a multivariate binary environment, i.e., $S = \prod_{j=1}^m S_j$ for fixed $m \in \mathbb{N}$ and $S_j = \{s_j, \bar{s}_j\}$, such that $S \cong \mathbb{B}^m$. This has the advantage of a compact representation; states can be written as Boolean arrays or encoded as natural numbers. I adopt this restriction for now, but keep in mind that my governance approach should, if possible, not be limited to this setting, but apply to (at least) arbitrary finite domains. I expect the permission of infinite or continuous domains or even irregular environmental “shapes” to pose new challenges, and will comment on this problem in Section 4.3.4.

Regarding actions and transitions, first assume that $\mathcal{A}_i \subseteq \{\emptyset, 1, \dots, m\}$ and

$$\delta(s, a) = s' \text{ where } s'_j = \begin{cases} \bar{s}_j & \text{if } \exists i : a_i = j \\ s_j & \text{else} \end{cases}$$

which means that agents can choose to change one attribute per time step (or to do nothing, by choosing the neutral action \emptyset), and each attribute is toggled if at least one agent chooses to change it. As above, allowing more general environmental structures and more complex actions and transitions would cause additional challenges and require some additional assumptions. For example, aggregating agent actions with respect to a non-binary attribute [23] can be complex in itself: Does an agent request a certain value for a numerical attribute, or does it request a certain offset? Is the new value simply the mean of all requested values? Does it maybe only change when the agents can agree on a new value?

Theorem 1. *Let \mathcal{M} be a GMAS with n agents, m binary attributes and q fundamental actions per agent. Then, for a given cost threshold $\alpha \geq c_G(\delta(s_t, \emptyset), \alpha)$, a Pareto-minimal restriction $\mathcal{A}_i \subseteq \bar{\mathcal{A}}$ can be computed in time $O(n^2 \cdot q^{(n+2)})$.*

Proof. See [30].

Note that the complexity of this algorithm does not depend on the size of the environment, as long as the past observed actions per state are readily available. Therefore, it is suitable for MAS with large state spaces, but few actions—a typical

scenario would be a video game where each player can take a constant (low) number of actions.

As shown in a first prototypical setup, it turns out that the smart home case (Example 2) can indeed be successfully governed by an algorithm built from Theorem 1. Figure 4.3 shows a part of the evaluation of [30] using a variable number of agents (2 = dotted line, 3 = dashed, 5 = continuous) which were set up with random state-action mappings and acted in this system for $0 \leq t \leq 100$. The chart shows a comparison between ungoverned and governed simulations, including the average cost for both simulations and the degree of restriction in the governed simulation. To minimise outliers, each line is the mean of 10 independent runs of the same simulation.

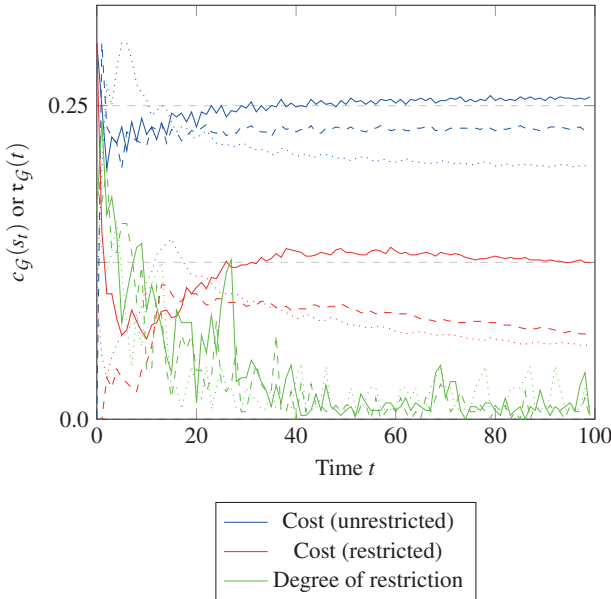


Fig. 4.3. Simulation of the Smart Home Example

4.3.3 Implementation

A meaningful evaluation of the theoretical approach is an essential ingredient for a PhD thesis which claims to provide a practical solution for governing competitive MAS. Great attention must thus be paid to an evaluation framework which allows for the testing of the approach as well as for a detailed comparison with competing approaches. Although the Governance component is the core of research and development, the overall performance and reliability also widely depend on realistic agents. If those agents are not immediately controlled by human players, there is still a need for

strategic and “intelligent” behaviour in order to validate (or invalidate) the capabilities of the Governance.

In order to provide an optimal environment for development and testing, I am developing a Python-based Multi-Agent framework, specifically designed to be fed with different agent and governance functionalities. This framework provides measuring, logging and analysis of performance as well as direct comparison of different governance approaches (including no governance at all).

4.3.4 Challenges, Refinements and Extensions

While the general setting is very broad and applies to a wide range of MAS, I have made some restrictions and neglected particular issues so far in order to reduce complexity. Some of the topics which haven’t been considered but are crucial for a deep understanding of Governed MAS are listed and explained in this section.

4.3.4.1 Fairness

In the current implementation (see Section 4.3.3), the Governance can effectively reduce its cost by defining restrictions based on an expected cost matrix. This approach forbids actions according to their expected cost impact, without taking into account previous restrictions or balancing the degree of freedom between agents. In extreme cases, the strategy can lead to some agents always being restricted to just one action, while others are not affected at all.

A natural question regarding this issue is whether “fairness” should be part of the Governance’s decision process or even part of the system-level objective. If so, the concept of fairness needs to be well-defined in the context of MAS, and the Governance must be given a means to distinguish restrictions with respect to their evenness.

4.3.4.2 Derivation of Rules

The restriction function Γ is not required to provide any consistency, i.e., there is no link between \mathcal{A}_t and \mathcal{A}_{t+1} apart from the fact that both are subsets of $\bar{\mathcal{A}}$. Consequently, agents cannot anticipate what restrictions will be posed on their action space in the future. At the same time, the Governance does not justify its decisions or provide any reasons for them, but merely states what it allowed at the current step.

It might be useful to derive explicit rules or criteria for restricted and allowed actions, which could be expressed in a formal language. This would allow for better analysis of a system, for example regarding the link between agent behaviour and rule emergence. The field of Explainable AI deals with a similar issue of deriving abstract knowledge from sub-symbolic data.

4.3.4.3 Open agent sets

A typical problem with MAS is that agents, as they are autonomous entities, cannot be forced to do something. This implies that an agent might not react at all when it is asked to choose an action, or it might respond with incomprehensible or illegal data. In Section 4.2.2, a neutral action was introduced to cater for this fact—simply assume this action to be substituted for any invalid agent response. Nevertheless, the problem of agents spontaneously entering or leaving the system raises another question: Should the Governance treat all agents independently and individually? It might be a good idea to have a model which can handle unknown agents and apply some “general knowledge” to them, instead of assuming an empty knowledge base for each new agent. Such an approach would on the one hand free the Governance from having to identify and track each agent separately, and on the other hand allow it to (partly) carry over its knowledge to new agents joining the system.

4.3.4.4 Dynamic Agent Goals

It cannot, in general, be expected that agents remain consistent in their goals over the run-time of the system. In contrast, it is reasonable to assume that goals and strategies change gradually (not abruptly) over time. Therefore, the Governance should incorporate a mechanism which can deal with changing goals, for example by discounting old observations, or by categorising former observations according to consistency with the latest observed actions. This line of reasoning is closely connected to the field of belief revision [13].

4.3.4.5 Structure of environments and actions

When the environment consists of binary attributes and actions are merely toggling single attribute values, an MAS is fairly well-arranged. This, however, does not always represent the reality: There can be continuous or entangled environmental states, complex actions, non-trivial aggregation rules for different actions, and other complications. While the concrete implementation of a governance algorithm most likely depends on the choice of such properties, its general applicability should range over as large a class of systems as possible, and thus be able to deal with the general model from Section 4.2 instead of just binary multi-attribute MAS.

4.3.4.6 Distributed Governance

Multi-Agent Systems are one of the most common form of distributed systems, in which the overall computation task is carried out by independent entities which do not require central control and not even global information. Since this is a major asset of such systems, it seems counterproductive to add a central Governance which needs to aggregate and evaluate all agent actions at every step in order to do its job.

As a consequence, I will look at parallelising the Governance in order to ensure scalability. It seems that much of its work can be executed in a map-and-reduce

fashion, but the existing algorithms haven't been designed according to this paradigm yet.

4.4 Related Work

The bulk of Multi-Agent research deals with the task of teaching agents how to act [17, 31], both in the cooperative case where there is a common goal and in the competitive case where conflicts are inherent. In contrast, I take the viewpoint of an outside entity wanting to “guarantee the successful coexistence of multiple programs” [37], that is, to define a degree of success and then influence it via suitable actions. Multi-Agent Systems can be classified with respect to this criterion as shown in Figure 4.4:

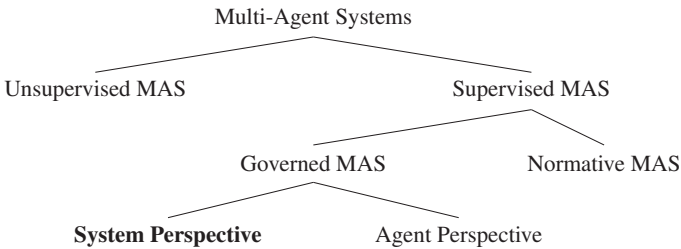


Fig. 4.4. Classification of Multi-Agent Systems

An MAS can either have a supervising entity which interferes with the agents in order to achieve a system objective, or this goal is achieved solely by the interaction of the agents (self-organisation and/or emergence [41], [26]).

When there is a supervisor, its decisions can be either binding (which I will call a *governed* MAS) or non-binding (normative). I follow here the reasoning of [4] who state that norms are “a concept of social reality [which does] not physically constrain the relations between individuals. Therefore it is possible to violate them.” Note that this terminology is far from being unambiguous; for instance, [29] use the term *Normative Synthesis* for the *enforcement* of equilibria.

There are two perspectives of a Governed MAS: The viewpoint of a participating agent and that of the governing instance. In the latter case, the key points of interest are the level of control (or level of satisfaction of the system objectives) that can be achieved, and the necessary intervention.

There are many approaches developed from an agent perspective which can partly be applied to the system point of view, e.g., opponent modelling and Multi-Agent reinforcement learning. However, only few areas (e.g. *Normative Multi-Agent Systems* [5]) have been thoroughly examined from an observer's angle.

[17] and [16] identify two main research streams for competitive Multi-Agent Learning: Game theoretic approaches including auctions and negotiations, and Multi-

Agent Reinforcement Learning [36]. The latter add a layer of complexity to classical reinforcement learning [9], since competitive agents all evolve at the same time and therefore disturb the learning process of their opponents (*moving-target problem*) [28]. Both surveys, however, restrict their scope to learning agents, instead of external entities learning *about* agents.

Game theory in this context oftentimes deals with small, well-defined (and mostly contrived) scenarios [3, 15, 38] like two-player games with a fixed payoff matrix, which can be formally examined and sometimes even completely solved in terms of optimal responses and behavioural equilibria. What these solutions lack is widespread applicability to real-world settings where information is incomplete, environments are large and agents do not behave predictably. Therefore, the gap between academic use cases on the one hand and industrial and societal applications on the other hand is still large.

[37] realised that social laws can be used by designers of Multi-Agent Systems to make agents cooperate without controlling the agents themselves. They describe an approach to define such laws off-line and keep them fixed for the entire runtime of the system, and they mention the possibility that their laws are not always obeyed by the agents. From this reasoning, the two notions of *hard norms* and *soft norms* [33, 34] have emerged—the two categories which I call Governed MAS (GMAS) and Normative MAS (NMAS), respectively [19].

[34] argue that “achieving compliance by design can be very hard” due to various reasons (e.g. norm consistency and complexity of enforcement). Therefore, they reach the conclusion that NMAS are more suitable for open and distributed environments. In turn, the lack of hard obligations leads to concepts like sanctions, norm revision, norm conflict resolution, and others. NMAS have been researched from various perspectives and with various theoretical frameworks, among them formal languages and logics [7, 12, 29], Bayesian networks for the analysis of effectiveness [11], bottom-up norm emergence [26], and online norm synthesis [25]. Many of these approaches are also partially applicable to Governed MAS, but require adaptation and generalisation.

Another well-known problem of MAS is scalability [17, 40], especially for large state spaces. While the number of states is obviously exponential in the number of environmental variables, reasonable additional assumptions about the dependencies between variables can lead to much more compact representations of knowledge regarding preferences and utilities. Famously, this reasoning has been applied in the development from Q-learning [39] to Deep Q-learning [24]. While Q-tables and the corresponding Neural Networks describe the expected payoff of an action at a given environmental state (from an agent perspective) and hence define the choice of the next action, I need to describe the probability distribution of an action set, given an environmental state (from an observer’s perspective).

Regarding preference orders over a set of alternatives, CP-nets [6] are among the most common data structures for encoding partial orders and enriching given knowledge with observations. They have been used extensively for preference aggregation [21, 32] and preference learning [8, 10, 14], both for general entities and in the Multi-Agent context. Allen [1] has extended the framework to finite attribute domains

and indifference, while others [2, 22] have tackled the problem of deriving total orders from a given CP-net.

Yet, those preference-based approaches represent orders over environmental states, while I need to describe orders over action spaces, depending on the value of the environmental attributes. Although these approaches cannot (as illustrated in Section 4.1.1) lead to accurate results in case of a discrepancy between observed and intended behaviour, they still have some interesting implications for the present scenario: First, they show how dependencies between attributes can be used to achieve a more compact and exploitable data structure. Second, the process of deriving knowledge about agent behaviour from observing them is similar (when preferences are not already assumed to be known, as in [11]), such that the use of an analogous structure seems a reasonable next step for my Governance approach.

The self-adaptivity and self-organisation properties of Multi-Agent Systems have been seen as related to Organic Computing Systems by several researchers [20]. The GMAS approach targets the conflicts stemming from differing agent goals and from lack of cooperation by introducing a mediating Governance instance. A similar line of thought was established in [41] in the context of self-organisation and the emergence of cooperation.

4.5 Application for Evaluation

The domain chosen for Example 2 lends itself on several levels to examination as a MAS with system objectives and subsequent need for governance: The agents can have conflicting goals and only express them by acting within the system, there are dependencies between agent actions, and there are undoubtedly undesirable states which should be avoided even if this requires restricting the agents. However, it lacks two more criteria which make an interesting case for an online self-learning Governance, especially as a proof-of-concept for the contribution of the PhD project—Safety-criticality and real-time requirements. Those criteria are satisfied by another application domain: Autonomous vehicles.

The current baseline for designing autonomous cars is that they have to obey the (static) local traffic rules, which includes the ability to detect anomalies and dangers and react accordingly. These regulations are identical for all road users and do not, in general, take into account any specific agent goals. As a consequence, avoiding traffic jams or shortages of parking space can only be addressed globally or via explicit human intervention.

I claim that a self-learning Governance which is given a set of objectives for an autonomous traffic scenario can achieve this to a high extent in an ad-hoc fashion while ensuring compliance with basic safety rules.

Since similar scenarios have been examined in related work, it should be possible to establish a well-defined baseline against which the performance of the GMAS approach can be measured.

4.6 Conclusion

In Multi-Agent research, there is a large gap between agent-centric and system-focused (or governance-focused) learning methods. While individual agents experience a lot of attention from the Game Theory, Logic and Machine Learning communities, governance (both centralised and distributed) leads more of a niche existence, and oftentimes the prerequisites regarding agent behaviour are very specific.

I am aiming towards closing this gap and advancing the area of Governed Multi-Agent Systems such that both effective and minimally restrictive governance becomes available for large and currently uncontrollable systems. To achieve this, formal models and efficient data structures are just as important as governance algorithms which can deal autonomously with incomplete information and unknown, ever-changing agent strategies.

References

1. Allen, T.E.: CP-nets with indifference. In: 2013 51st annual allerton conference on communication, control, and computing (allerton). pp. 1488–1495 (2013)
2. Aydogan, R., Baarslag, T., Hindriks, K., Jonker, C., Yolum, P.: Heuristic-Based Approaches for CP-Nets in Negotiation. In: Studies in Computational Intelligence, vol. 435, pp. 113–123 (Jan 2013), journal Abbreviation: Studies in Computational Intelligence
3. Bade, S.: Nash Equilibrium in Games with Incomplete Preferences. *Economic Theory* 26(2), 309–332 (2005), www.jstor.org/stable/25055952, publisher: Springer
4. Balke, T., da Costa Pereira, C., Dignum, F., Lorini, E., Rotolo, A., Vasconcelos, W., Villata, S.: Norms in MAS: Definitions and Related Concepts (Jan 2013), pages: 31
5. Boella, G., van der Torre, L., Verhagen, H.: Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory* 12(2), 71–79 (Oct 2006), <https://doi.org/10.1007/s10588-006-9537-7>
6. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-Nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Int. Res.* 21(1), 135–191 (Feb 2004)
7. Bulling, N., Dastani, M.: Norm-based Mechanism Design. *Artif. Intell.* 239(C), 97–142 (Oct 2016), <https://doi.org/10.1016/j.artint.2016.07.001>
8. Chevalere, Y., Koriche, F., Mengin, J., Zanuttini, B.: Learning Ordinal Preferences on Multiattribute Domains: the Case of CP-Nets. *Preference Learning* (Jan 2011)
9. Claus, C., Boutilier, C.: The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. pp. 746–752. AAAI '98/IAAI '98, American Association for Artificial Intelligence, Menlo Park, CA, USA (1998), <http://dl.acm.org/citation.cfm?id=295240.295800>
10. Cornelio, C., Goldsmith, J., Mattei, N., Rossi, F., Venable, K.B.: Updates and Uncertainty in CP-Nets. In: Cranefield, S., Nayak, A. (eds.) *AI 2013: Advances in Artificial Intelligence*. pp. 301–312. Springer International Publishing, Cham (2013)
11. Dell'Anna, D., Dastani, M., Dalpiaz, F.: Runtime Revision of Norms and Sanctions Based on Agent Preferences. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 1609–1617. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2019), event-place: Montreal QC, Canada

12. García-Camino, A., Rodríguez-Aguilar, J., Sierra, C., Vasconcelos, W.: A rule-based approach to norm-oriented programming of electronic institutions. *SIGecom Exchanges* 5 (Jan 2006)
13. Gärdenfors, P.: Belief Revision: an introduction. In: *Belief Revision* (May 1992), journal Abbreviation: *Belief Revision*
14. Guerin, J.T., Allen, T.E., Goldsmith, J.: Learning CP-net Preferences Online from User Queries. In: Perny, P., Pirlot, M., Tsoukiàs, A. (eds.) *Algorithmic Decision Theory*. pp. 208–220. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
15. Gutierrez, J., Perelli, G., Wooldridge, M.: Imperfect information in reactive modules games. *Information and Computation* 261, 650 – 675 (2018)
16. Hernandez-Leal, P., Kartal, B., Taylor, M.: A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* (Oct 2019)
17. Hoen, P.J.t., Tuyls, K., Panait, L., Luke, S., La Poutré, J.A.: An Overview of Cooperative and Competitive Multiagent Learning. In: Tuyls, K., Hoen, P.J., Verbeeck, K., Sen, S. (eds.) *Learning and Adaption in Multi-Agent Systems*. pp. 1–46. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2006)
18. Jennings, N.R., Wooldridge, M.J.: *Agent technology: foundations, applications, and markets*. Springer Science & Business Media (2012)
19. Kantert, J., Edenhofer, S., Tomforde, S., Hähner, J., Müller-Schloer, C.: Normative control: Controlling open distributed systems with autonomous entities. In: *Trustworthy Open Self-Organising Systems*, pp. 89–126 (2016)
20. Krupitzer, C., Breitbach, M., Roth, F.M., VanSyckel, S., Schiele, G., Becker, C.: A survey on engineering approaches for self-adaptive systems (extended version) (2018), <https://madoc.bib.uni-mannheim.de/44034/>
21. Kyaw, H., Ghosh, S., Verbrugge, R.: Multi-player multi-issue negotiation with mediator using CP-nets. *ICAART 2013 - Proceedings of the 5th International Conference on Agents and Artificial Intelligence* 1, 99–108 (Jan 2013)
22. Lang, J., Mengin, J.: The complexity of learning separable ceteris paribus preferences. In: *Proceedings of the 21st international joint conference on artificial intelligence*. pp. 848–853. IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009)
23. List, C.: Social choice theory. In: Zalta, E.N. (ed.) *The Stanford encyclopedia of philosophy*. Metaphysics Research Lab, Stanford University, winter 2013 edn. (2013), <https://plato.stanford.edu/archives/win2013/entries/social-choice/>
24. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* 518, 529–33 (Feb 2015)
25. Morales, J.: *On-line norm synthesis for open Multi-Agent systems*. Ph.D. thesis, Universitat de Barcelona (2016)
26. Morris-Martin, A., De Vos, M., Padget, J.: Norm emergence in multiagent systems: a viewpoint paper. *Autonomous Agents and Multi-Agent Systems* 33(6), 706–749 (Nov 2019), <https://doi.org/10.1007/s10458-019-09422-0>
27. Müller-Schloer, C., Tomforde, S.: *Organic Computing - Technical Systems for Survival in the Real World*. Birkhäuser (2017)
28. Nowé, A., Vrancx, P., De Hauwere, Y.M.: Game Theory and Multi-agent Reinforcement Learning. In: Wiering, M., van Otterlo, M. (eds.) *Reinforcement Learning: State-of-the-Art*, pp. 441–470. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-27645-3_14

29. Perelli, G.: Enforcing Equilibria in Multi-Agent Systems. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 188–196. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2019), event-place: Montreal QC, Canada
30. Pernpeintner, M.: Toward a self-learning governance loop for competitive multi-attribute mas (submitted)
31. Rizk, Y., Awad, M., Tunstel, E.: Decision Making in Multi-Agent Systems: A Survey. *IEEE Transactions on Cognitive and Developmental Systems* PP, 1–1 (May 2018)
32. Rossi, F., Venable, K., Walsh, T.: mCP Nets: Representing and Reasoning with Preferences of Multiple Agents. (Jan 2004), journal Abbreviation: Proceedings of the National Conference on Artificial Intelligence Pages: 734 Publication Title: Proceedings of the National Conference on Artificial Intelligence
33. Rotolo, A.: Norm compliance of rule-based cognitive agents. pp. 2716–2721. *IJCAI International Joint Conference on Artificial Intelligence* (Jan 2011)
34. Rotolo, A., van der Torre, L.: Rules, Agents and Norms: Guidelines for Rule-Based Normative Multi-Agent Systems. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) *Rule-Based Reasoning, Programming, and Applications*. pp. 52–66. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
35. Russell, S., Norvig, P.: *Artificial intelligence: A modern approach*. Prentice Hall Press, USA, 3rd edn. (2009)
36. Shoham, Y., Powers, R., Grenager, T.: *Multi-Agent Reinforcement Learning: a critical survey* (Jun 2003)
37. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. *Artificial Intelligence* 73(1), 231 – 252 (1995), <http://www.sciencedirect.com/science/article/pii/000437029400007N>
38. Stirling, W.C., Felin, T.: Game theory, conditional preferences, and social influence. *PLOS ONE* 8(2), 1–11 (Feb 2013), <https://doi.org/10.1371/journal.pone.0056751>
39. Watkins, C.: *Learning From Delayed Rewards* (Jan 1989)
40. Weyns, D., Michel, F.: Agent environments for multi-agent systems – a research roadmap. In: Weyns, D., Michel, F. (eds.) *Agent environments for multi-agent systems IV*. pp. 3–21. Springer International Publishing, Cham (2015)
41. Wolf, T.D., Holvoet, T.: Emergence and self-organisation: a statement of similarities and differences (2004)

Interpolated Experience Replay - A Roadmap

Wenzel Pilar von Pilchau

OrcidID  0000-0001-9307-855X

Organic Computing Group

University Augsburg

Wenzel.Pilar-von-Pilchau@informatik.uni-augsburg.de

Abstract. Reinforcement Learning and especially Deep Reinforcement Learning are fields of great interest: (mathematical) interpolation is used to get information of data points in an area where only neighbouring samples are known: This could be a promising extension for the Experience Replay, which is a major component of deep Reinforcement Learning. Interpolating experiences stored in the Experience Replay could speed up learning in the early phase and reduce the amount of exploration needed. A first simple approach of averaging rewards in a setting with unstable transition function and very low exploration is implemented. The evaluation of this initial approach has shown promising results and warrants more detailed research.

Keywords: Machine Learning, Organic Computing, Deep Reinforcement Learning, Deep Q-Network, Experience Replay, Interpolation.

5.1 Motivation

Reinforcement Learning (RL) [24], as one of the big three Machine Learning (ML) [13] domains next to Supervised and Unsupervised Learning, bears great potential on the road towards an Artificial Intelligence. RL already showed great success in (video-) games [14, 21, 27] and achieved 'super human performance'. An important component of a lot of these algorithms is the Experience Replay (ER). Some Deep Reinforcement Learning (DRL) algorithms even rely on it to work properly [15], and others use it to improve their efficiency [9].

The ER, in its simplest form, represents a storage of experienced transitions of the agent. These memories can then be used to improve sample efficiency or, through a uniform sampling over the data, to remove the correlation of successive transitions. Some more experienced versions [1, 19] alter the sampling probability or include some kind of synthetic experiences to support the learning process of the agent.

The Concept of the Interpolated Experience Replay follows the idea of inducing synthetic experiences by means of interpolation. In the long run the approach should speed up the learning of a value function [24] in DRL algorithms.

The rest of the paper is organized as follows: Section 5.2 gives an introduction to some basic knowledge and arranges the intended research contribution in the context of Organic Computing. An overview of work related to this approach in Section 5.4.3 is followed with a sketched overview of the research agenda in Section 5.4. Finally, the paper is closed in Section 5.5

5.2 Background

This chapter gives a short reminder of some Organic Computing Concepts and how the presented approach fits into this domain. Next a short introduction into the theoretical basics of relevant research areas is given. Namely these are Reinforcement Learning, Experience Replay, Deep Reinforcement Learning and Interpolation.

5.2.1 Organic Computing

Organic Computing (OC) [16,25] describes the design of “life-like” technical systems with so-called *self-* properties*. An OC system therefore has the ability to act based on its own decisions. A related topic is Autonomic Computing (AC), which uses the biological principle of the autonomic nervous system as paradigm. Each OC system is equipped with sensors (to observe the environment) and actuators (to act on it). It adapts autonomously to the environment description received from its observer. The reaction has to be in a way that the system remains functional. To fulfil this requirement, even in, yet unseen states and unanticipated conditions, an OC system is typically based on (machine) learning.

The identified research questions (Section 5.4.3) concerns machine learning in general, and in particular RL, and so the presented approach fits the context of OC very well. Also, ER is a nature inspired technique/concept. Due to its relationship to Deep RL it lies within the scope of OC as well.

An example how the presented approach could be implemented in a real world scenario would be a robot solving a maze. This robot has the ability to observe its surroundings and can use this information to create a description of the current state. Also it is able to move and therefore can change the state via its actions. Internally it saves all the experienced state-transitions and can use these to create synthetic ones via interpolation (this can somehow be compared with imagination of humans). These synthetic experiences, together with the real ones, are then utilised to learn a policy for reaching the exit of the maze.

5.2.2 Reinforcement Learning

This section is based on Sutton and Barto’s: “Reinforcement Learning: An Introduction” [24].

In general, RL can be seen as learning what to do. In a more precise way, this means learning how to map situations to actions. A RL learner, also called agent,

gets a numerical reward signal after executing an action and his task is to maximize this signal. The agent has no initial knowledge about which action it should take, but instead must discover which action returns the highest reward by trying them. This trial-and-error search is called exploring and forms, together with the delayed reward, the most important characteristics of RL. The latter explains the fact that an action not only affects the immediate reward, but also has an impact on the next situation and consequential all successional rewards.

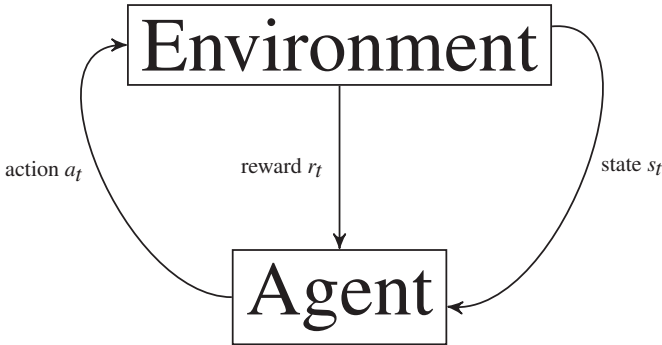


Fig. 5.1. The Reinforcement Learning Loop

In Figure 5.1 a basic RL scenario can be seen. It consists of an agent which interacts with an environment. At a time step t the agent gets the information of the state s_t he is actual in from the environment. On basis of this information the agent can calculate an action a_t he will perform in this state. After he has decided, he sends the information to the environment, it calculates the reward r_t that a_t returns and also the next state s_{t+1} the agent ends up in. Both are send to the agent and it can now perform a learning step with the transition 4-tuple (s_t, a_t, r_t, s_{t+1}) . This loop repeats until the agent has converged or a stop criterion is reached.

On challenge in RL, which was already mentioned above, is the explore-exploit dilemma. This describes the fact, that an agent trying to maximize its reward must prioritize actions it has already tried in the past and found them to be effective in generating reward. But to discover such actions, it has to try new - never chosen before - actions. The agent has to *exploit* the knowledge he has already gained to gather reward, but it also has to *explore* in order to make better decisions in the future. The dilemma is that neither exploration nor exploitation can be executed exclusively without failing at the task. The agent has to combine both to succeed.

The main subelements - beyond the recently discussed agent and environment - of a RL system are the following four: a *policy*, a *reward signal*, a *value function*, and, optionally, a *model* of the environment. We will take a deeper look at the first three ones.

The way a learning agent behaves at a given time is called its *policy*. More precise: the agent's policy decides which action it takes in which state. Roughly speaking

one could imagine the policy as a mapping of its to actions. The policy is the core element of a RL agent as it alone is sufficient to determine behavior. A policy may be stochastic, specifying probabilities for each action. The *reward signal* has already been mentioned above and defines the goal of a RL problem. To determine which events are good and which are bad, the agent uses the reward received from the environment. The agent's objective over the long run is to maximize the total reward he receives and therefore these signals are the main component for the agent to alter its policy. Contrary to this immediate measure of what is good, the *value function* specifies what is good on the long run. The value of a state is the total reward an agent can expect to accumulate over the future, starting in the particular state and following its policy. For example, a state with a low immediate reward might have a high value, because it is followed by other its which yield a high reward.

Many of the core algorithms of RL are inspired by biological learning systems, and therefore RL is - compared to other forms of ML - the closest to the kind of learning humans and other animals perform.

5.2.3 Experience Replay

The Experience Replay [10, 11] was introduced by Lin to increase sample efficiency and speed up convergence in RL. It is a biological inspired mechanism [12, 17] and can be understood as a collection of previous experiences the agent has made up to this point. One experience is therefore defined as $e_t = (s_t, a_t, r_t, s_{t+1})$ and at each time step t the agent stores its recent experience in a data set $D_t = \{e_1, \dots, e_t\}$. This procedure is repeated over many episodes, where the end of an episode is defined by a terminal state. In the training phase, the agent uses every transition, including a policy action, stored in the ER for training. A policy action is defined as an action the agent would choose for a state, if it follows the current policy, with a certain probability above a threshold P_t . This technique was used together with so-called "connectionist" implementations of Q-Learning [24] and Adaptive Heuristic Critic [2], which describes an implementation using non-deep neuronal networks. The use of an ER increases the sample efficiency and therefore the learning speed. It is very easy to implement in its basic form and the cost of using it is mainly specified by the storage space needed to store it. [10]

5.2.4 Deep Reinforcement Learning

The following section is based on the publication "Playing Atari with Deep Reinforcement Learning" [14] and the dedicated article "Human-level control through deep reinforcement learning" [15]. Both were published by DeepMind Technologies in the Nature magazine.

Classical RL agents achieved some good results in a variety of domains, but are limited by the feasibility of the state representation. Successful domains are those where whether useful features can be handcrafted, or which are fully observable and of low-dimensionality. Recent advances in deep learning, especially in computer vision, made it possible to extract high-level features from raw sensory data and

produced better representations than handcrafted features. These breakthroughs led to a combination of RL and a class of artificial neural network called deep Q-network (DQN) [14], which operates directly on RGB images. This algorithm was able to successfully learn policies from high-dimensional sensory inputs using end-to-end reinforcement learning. The DQN utilized its neural network to approximate a Q-function in the form of:

$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \mid s_t = s, a_t = a, \pi \right]$$

This function represents the maximum expected sum of rewards r_t , discounted by γ at each time step t , achievable by a policy $\pi = P(a|s)$, after making an observation s and taking an action a .

A neural network is said to be a nonlinear function approximator. Unfortunately RL is known to be unstable or even diverge when the Q function is approximated by such [26] because of the following reasons:

1. the correlations present in the sequence of observations
2. the fact that small updates to Q may significantly change the policy and therefore change the data distribution
3. the correlations between the action-values and the target values

To address these issues an ER is used. In contrast to the basic ER from Lin described above, where all experiences with policy actions were used, Q-learning updates are applied on samples (or minibatches) of experience $(s, a, r, s') \sim U(D)$, which are drawn uniformly at random from the ER. This randomization breaks the correlation of the observations and succeeds in reducing the variance of the updates. Another advantage is that each experience is potentially used in more than one update, which results in a greater data efficiency. Finally, in an on-policy [24] learning scenario, current parameters determine the next data sample that the parameters are trained on. This can result in the occurrence of unwanted feedback loops and the parameters getting stuck in a poor local minimum, or even diverging catastrophically. ER prevents that through averaging the behaviour distribution over many of its previous states and smoothing out learning.

A deep CNN is used to parametrise an approximate value function $Q(s, a; \theta)$ where θ are the weights of the network. The performed Q-learning update uses the following loss function:

$$L(\theta) = \mathbb{E}_{(s, a, r, s') \sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta) \right)^2 \right] \quad (5.1)$$

This corresponds to the squared error of the Q value minus the target value where the target value is expressed as $r + \gamma \max_{a'} Q(s', a'; \theta)$. The same parameters are used for the Q and also the target value. In every weight update the parameters change. Subsequently increasing $Q(s_t, a_t)$ often also increases $Q(s_{t+1}, a)$ for all a and therefore increases the target value. This could lead to oscillations or divergence of the policy. To counteract this issue a second network, called target network, is introduced, which is only used for calculating the target values. The Q network is copied every C steps

to generate the target network. This procedure adds a delay between an update to Q and the time the update affects the targets, making divergence or oscillations much more unlikely.

5.2.5 Interpolation

In the process of approximating a function f , one tries to find another function φ which minimizes the distance relative to a norm $\|\cdot\|$. As there exist several functions φ which approximate f unequally well, $\hat{\varphi}$ is denoted as the best approximation for which holds:

$$\|f - \hat{\varphi}\| \leq \|f - \varphi\| \text{ for all } \varphi. \quad (5.2)$$

To find an approximation some knowledge about the function f is required. This knowledge is represented by a set of already seen data instances, more often denoted as sampling points $SP := (x_i, f(x_i))$. The function f is either defined on a finite interval $s_i \in [a, b]$ or a finite set of points $s_i \in SP$. [6]

If the best approximation $\hat{\varphi}$ in the discrete case holds the following condition:

$$\|f - \hat{\varphi}\| = 0 \quad (5.3)$$

then it also satisfies:

$$\hat{\varphi}(s_i) = f(s_i) \text{ for } i = 1, 2, \dots, N \quad (5.4)$$

This special case is called *Interpolation* and describes the state in which the best approximation $\hat{\varphi}$ is equal to the function f in every sampling point. To map a value to a new and unknown query point x_q , one can now approximate the interpolant between surrounding sampling points using low-degree polynomials. This is reasonable because a sufficiently small interval can be approximated arbitrarily well by low degree polynomials, even degree 1, or zero. Following this approach, not every sampling point has to be taken into consideration for calculating the query point's function value $\hat{\varphi}(x_q)$. It is possible to include all sampling points for calculation. This leads to a classification into to categories of interpolation: global methods which satisfy the latter and local methods which refer to the former. [6, 23]

One simple technique is the rather naive *Nearest Neighbour Interpolation*. It belongs to the distance-based approaches, which, in more detail, explains the fact, that the influence of each considered sampling point is directly related to the distance to the query point x_q . This technique searches for the sampling point s_i with the smallest distance to the query point x_q and assigns the associated function value $f(s_i)$ to $\hat{\varphi}(x_q)$. More formally, this can be expressed as follows:

$$s_{nearest} = \arg \min_{s_i} d(x_q, s_i)_2 \quad (5.5)$$

$d(x_q, s_i)_2$ denotes the L2-norm, also known as the Euclidean norm. Equation (5.5) states that $s_{nearest}$ is the coordinate of the sampling point located nearest to x_q . From this follows the interpolation equation:

$$\hat{\varphi}(x_q) = f(s_{nearest}) \quad (5.6)$$

Because only one sampling point is taken into account, the method results in a step-wise and non-smooth interpolation. This can be seen in Equation (5.6), as the function value $\hat{\phi}(x_q)$ does not change until another sampling point has a smaller distance to x_q and is therefore denoted as $s_{nearest}$. This algorithm is of low complexity and therefore cost-efficiently and easy implemented. [3, 23]

A more complex but also more accurate method represents the inverse distance weighting method from Shepard [20]. This technique uses more than one sampling point and therefore establishes a smoother interpolant.

5.3 Related Work

The classical ER, introduced in Section 5.2.4, is a basic, non-optimised technique, which has been improved in many further publications. One prominent improvement is the Prioritized Experience Replay [19], which replaces the uniform sampling with a weighted sampling in favour of samples which might influence the learning process the most. This induces bias and to correct this, importance-sampling has to be used. The authors show that a prioritised sampling improves the learning efficiency. Because replays store a lot of experiences and the sampling occurs in every training step, it is crucial to reduce the computational cost to a minimum.

De Bruin et al. [5] investigated the composition of samples in the ER. They discovered that for some tasks those transitions are important, that are created during the exploration-heavy early stage: they prevent overfitting. Therefore they split the ER in two parts, one with samples from the beginning and one with samples from the current training process. They also show that the composition of the data in an ER is vital for the stability of the learning process and at all times diverse samples should be included. This work examines the impact of virtual experiences in an ER, but instead of interpolation they use a simulator.

Jiang et al. investigated Experience Replays combined with model-based RL and implemented a tree structure to represent the transition and reward function [8]. In their research they trained a model of the problem and invented a tree structure to represent it. With this model they could simulate virtual experiences which they used in the planning phase to support learning. To increase sample efficiency, experiences are stored in an ER. This approach has some similarities to interpolation but addresses other aspects.

An interpolation of on-policy and off-policy model-free Deep Reinforcement Learning techniques is presented by Gu et al. [7]. Interpolation between on- and off-policy gradient mixes likelihood ratio gradient with Q-Learning. This provides unbiased, but high-variance gradient estimations.

Stein et al. use interpolation in combination with eXtended Classifier Systems to speed up learning in single-step problems [23]. They introduce a so-called Interpolation Component (IC), which this publication uses as basis for its interpolation tasks.

5.4 Interpolated Experience Replay

This section gives an introduction of the idea behind the Interpolated Experience Replay and also introduces the Interpolation Component. After that the identified research questions are presented and followed by an overview of the planned research agenda.

5.4.1 Interpolation Component

For implementing the interpolation, the Interpolation Component from Stein et al. [22, 23] is used as a basis. It consists of a Machine Learning Interface MLI, an Interpolant, an Adjustment Component, an Evaluation Component and the Sampling Points SP as shown in Figure 5.2. If the MLI decides it wants to alter its collection of sampling points, the appropriate sample s^* is handed to the Adjustment Component, there, following a decision function A, it is added to or removed from SP. If an interpolation is required, the Interpolation Component fetches required sampling points from SP and computes, depending on an interpolation technique I, an output o_{int} . The Evaluation Component provides a metric E to track a so-called trust-level T_{IC} .

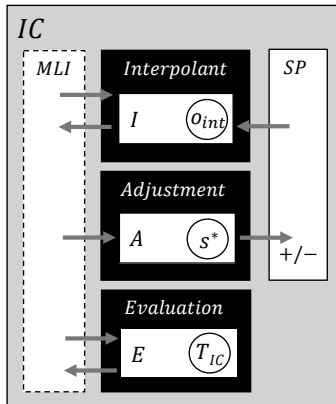


Fig. 5.2. The Interpolation Component from Stein et al. [23]

The ER replaces SP and consists of a queue with a maximum length and FiFo insert policy. This queue represents the standard ER and is filled only with real experiences, to store the synthetic samples another queue, a so-called *ShrinkingMemory*, is introduced. This second storage is of decreasing size, starting at a maximum it gets smaller depending on the length of the real valued queue. The *Interpolated Experience Replay* (IER) has a total size of the added length of both queues as can be seen in Figure 5.3 and also a defined maximum size. If this size is reached, the length of the ShrinkingMemory is decreased and the oldest items are removed, until either the real

valued queue gets to its maximum length and there is some space left for interpolated experiences or the IER fills up with real experiences.

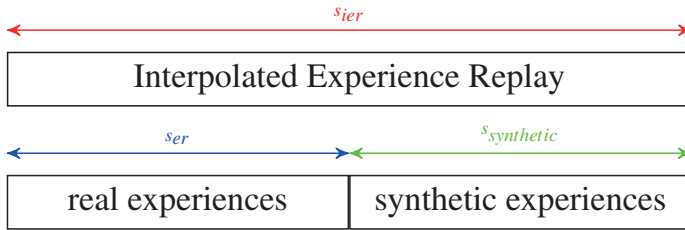


Fig. 5.3. The Interpolated Experience Replay

5.4.2 Idea

The underlying idea of the IER is to utilise stored real transitions to create synthetic experiences with interpolation. The storage can be seen as a kind of gathered knowledge about the problem space and in contrast to model-based RL [24], where the agent tries to learn the model and then use it for planning, IER takes another, but related, route and creates synthetic experiences to support the learning phase.

This method promises to improve the DQN algorithm, or similar approaches. Those comprise all methods where the neural network output persists of one node for every possible action, but only the chosen action is updated and all the other ones are not considered. This stems from the absence of knowledge of the corresponding reward and followup state from all but the chosen action. These missing values could be interpolated and in combination an update would consider all actions and therefore could speed up learning.

As the quality of the interpolation depends on the amount of gathered knowledge, it will grow over time. An indicator of the actual quality estimation at the time of the interpolation, e.g. a metric of the accuracy, could be stored together with the interpolation and then be used for the adjustment of the training update.

To realise the mentioned goals, a stepwise approach is planned, which is explained after the presentation the identified research questions in the next section.

5.4.3 Research Questions

To the following research question where identified:

5.4.3.1 RQ1

How can the gathered knowledge in the ER (in combination with interpolation) be utilised to assist the training process in Deep RL?

5.4.3.2 RQ2

Which parts of an experience can be created in a synthetic way and how? Which parts can be interpolated?

5.4.3.3 RQ3

How do synthetic experiences impact the performance of Deep RL?

5.4.4 Research agenda

Four different but related steps to answer the identified research questions are presented. The steps built up on one another.

5.4.4.1 Reward Averaging

Instead of creating synthetic rewards and followup states together, the initial focus was on interpolating only the rewards and using actual seen followup states. The approach was limited to discrete environments with discrete states. An unstable state-transition function results in a stochastic behaviour and makes the problem hard to solve. An action that started in one state will receive different rewards, depending of the next state the agent ends up in, but the amount of possible followup states is discrete. Based on stored experiences with the same start state and action, a synthetic reward could be calculated, using simple averaging as a very basic form of interpolation. A whole synthetic experience was created for every followup state, that was known to be reachable.

This has already be done on the *FrozenLakeEnv* from OpenAI Gym [4]. [18] shows that our approach can significantly increase learning efficiency in unstable, discrete environments. This serves as a proof of concept and laid the foundation for all the following research.

5.4.4.2 Interpolation of followup states

As “reward averaging” is limited to discrete and unstable environments, the next step includes the interpolation of followup states, so that IER can solve continuous environments as well. Several interpolation techniques (nearest neighbour, k-nearest neighbour, inverse distance weighting, etc. [20]) will be used to create synthetic values for the reward as well as the followup state.

In a first iteration, simple environments like e.g. *MountainCar* [4] will be used to test the concept and identify weaknesses. Also, interesting aspects that are worth a deeper investigation can be identified here. A second iteration will then examine more complex environments like e.g. Atari [4] inputs and spatial data. To use these states for interpolation in a realistic and result-oriented way, some pre-processing will be necessary. A possible solution could be dimension reduction and/or auto-encoders.

5.4.4.3 Parallel training of all possible actions

The last state of development would then be the simultaneous training of all possible actions in a DQN. One problem to solve is the quality estimation metric. After every step, when the agent gets the real values of the transition. The actual values could be compared to the values that would be produced by an interpolation with the real valued states as query points. The resulting error could hold for a starting point. But this procedure can lead to new challenges like e.g. how to transfer this accuracy to other transitions, where less stored data is available. Another question of efficiency: To what degree does this approach increase resp. decrease the performance?

5.4.4.4 Model-based IER

After the implementation and evaluation of the final concept, another extension that could be explore is the exchange of the interpolated followup states through states that are predicted by a learned state-transition function. This reminds of model-based RL, and one question that needs to be answered is: Does the positive effect outperform the additional costs of learning another function? However, this approach could handle complex state spaces utilising default techniques (e.g. Recurrent Neural Networks) instead of applying dimension reduction methods before the interpolation.

5.5 Conclusion

This work outlines the idea of the combination of methods from the domain of Deep RL, especially the experience replay and interpolation. After a short introduction of background for these topics, a general roadmap is described. First aspects have already been implemented and evaluated, but most still have to be investigated. In more detail: The averaging of rewards performed very well in discrete and non-deterministic environments. But this approach is also limited to these kind of environments. Nevertheless the results give a proof of concept. The presented combination is thought to bear great potential for speeding up the training process in Deep RL.

References

1. Andrychowicz: Hindsight experience replay. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 30, pp. 5048–5058. Curran Associates, Inc. (2017), <http://papers.nips.cc/paper/7090-hindsight-experience-replay.pdf>
2. Barto, A.G., Sutton, R.S., Watkins, C.J.C.H.: *Learning and sequential decision making*. In: *LEARNING AND COMPUTATIONAL NEUROSCIENCE*. pp. 539–602. MIT Press (1989)
3. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: *Orthogonal range searching: Querying a database*. *Computational Geometry: Algorithms and Applications* pp. 95–120 (2008)

4. Brockman, G., Cheung, L., Pettersson, J., Schneider, J., Schulman, J., Tang, W., null Zaremba: Openai gym (2016)
5. De Bruin, T., Kober, J., Tuyls, K., Babuška, R.: The importance of experience replay database composition in deep reinforcement learning. In: Deep reinforcement learning workshop, NIPS (2015)
6. Gautschi, W.: Approximation and interpolation. In: Numerical Analysis, pp. 55–158. Birkhäuser Boston, Boston (2012), http://dx.doi.org/10.1007/978-0-8176-8259-0_2
7. Gu, S.S., Lillicrap, T., Turner, R.E., Ghahramani, Z., Schölkopf, B., Levine, S.: Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 3846–3855 (2017)
8. Jiang, W., Hwang, K., Lin, J.: An experience replay method based on tree structure for reinforcement learning. IEEE Transactions on Emerging Topics in Computing pp. 1–1 (2019)
9. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
10. Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning 8(3), 293–321 (May 1992), <https://doi.org/10.1007/BF00992699>
11. Lin, L.J.: Reinforcement learning for robots using neural networks. Tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE (1993)
12. McClelland, J.L., McNaughton, B.L., O'Reilly, R.C.: Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review 102(3), 419 (1995)
13. Mitchell, T.M., et al.: Machine learning. 1997. Burr Ridge, IL: McGraw Hill 45(37), 870–877 (1997)
14. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing atari with deep reinforcement learning. CoRR abs/1312.5602 (2013), <http://arxiv.org/abs/1312.5602>
15. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529 (2015)
16. Müller-Schloer, C., Tomforde, S.: Organic Computing - Technical Systems for Survival in the Real World. Birkhäuser (2017), <https://doi.org/10.1007/978-3-319-68477-2>
17. O'Neill, J., Pleydell-Bouverie, B., Dupret, D., Csicsvari, J.: Play it again: reactivation of waking experience and memory. Trends in Neurosciences 33(5), 220 – 229 (2010), <http://www.sciencedirect.com/science/article/pii/S0166223610000172>
18. von Pilchau, W.B.P., Stein, A., Hähner, J.: Bootstrapping a dqn replay memory with synthetic experiences. arXiv preprint arXiv:2002.01370 (2020)
19. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. CoRR abs/1511.05952 (2015), <http://arxiv.org/abs/1511.05952>
20. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM National Conference. pp. 517–524. ACM '68, ACM, New York, NY, USA (1968), <http://doi.acm.org/10.1145/800186.810616>

21. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. *Nature* 529, 484 (Jan 2016), <https://doi.org/10.1038/nature16961>
22. Stein, A., Rauh, D., Tomforde, S., Hähner, J.: Architecture of Computing Systems – ARCS 2016: 29th International Conference, Nuremberg, Germany, April 4-7, 2016, Proceedings, chap. Augmenting the Algorithmic Structure of XCS by Means of Interpolation, pp. 348–360. Springer International Publishing, Cham (2016), http://dx.doi.org/10.1007/978-3-319-30695-7_26
23. Stein, A., Rauh, D., Tomforde, S., Hähner, J.: Interpolation in the extended classifier system: An architectural perspective. *Journal of Systems Architecture* 75, 79–94 (2017)
24. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
25. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic computing in the spotlight. *CoRR abs/1701.08125* (2017), <http://arxiv.org/abs/1701.08125>
26. Tsitsiklis, J.N., Van Roy, B.: Analysis of temporal-difference learning with function approximation. In: *Advances in neural information processing systems*. pp. 1075–1081 (1997)
27. Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W.M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., Silver, D.: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/> (2019)

Opportunistic Knowledge Adaptation in Self-Learning Systems

Simon Reichhuber

OrcidID  0000-0001-8951-8962

Intelligent Systems, Christian-Albrechts-Universität zu Kiel

sir@informatik.uni-kiel.de

Abstract. In the context of Autonomous Learning, the question arises how an online learning system adapts its knowledge according to a changing environment, i.e. arrival of new classes or changing noise functions, to maintain a robust level of performance. As a solution, we suggest an architectural design inspired by a variant of the Observer/Controller framework. Here, a working unit is guided by a control mechanism which implements the Observer/Controller pattern. The control mechanism itself can be represented by a classical Machine Learning model, either a static model (Support Vector Machine, Decision Tree) step-by-step retrained for a fixed state of available knowledge, or an Online Learning model (Learning Classifier System, Q-Learning) continuously improving its performance. Having aggregated feedback values from the environment about the quality progressing, a superior layer judges about the learning and occasionally replaces the static training set, the learning model itself, or both. We present a scenario, in which the presented architecture is assumed to improve the performance, because the system is aware of currently available knowledge and can opportunistically exploit this knowledge.

Keywords: Organic Computing, Online learning, Autonomous Learning, Knowledge Adaptation

6.1 Introduction

In the year 1993, Marc Weiser postulated his vision about a future in which the number of devices and their degree of connectivity explodes [20]. He claimed that the interfaces should vanish unobtrusively into background. Later in 2003, the Autonomic Computing initiative [8] and 2004 the Organic Computing initiative [11] described blueprints for systems that are aware of themselves and their environment.

Driven by these general concepts, the last two decades have witnessed a strong trend towards autonomous systems, i.e. a shift of responsibilities from engineers to the systems themselves. As a reaction to the need of finding appropriate reactions to unanticipated conditions, changing system constellations, and disturbances or even attacks, the technical systems are increasingly equipped with capabilities to self-adapt their behaviour, to identify new strategies, and to alter their integration status in the

overall system structure. Such a process requires some kind of creativity – which is realised by machine learning techniques.

The basis for these developments lies in the sub-domain of machine learning that focuses on learning at runtime without the need of, e.g. large amounts of training data and is called Autonomous Learning (AL). By now, most of the self-adaptation and self-organising systems (SASO) have made use of a dedicated and problem-based usage of specific machine learning techniques. However, the basic idea of this PhD project is that this is not sufficient for dynamically changing problems, open environments and hidden mutual influences among the behaviour of autonomous subsystems. Consequently, the main research question of this PhD project is the following

“How can we improve the utilisation of different dynamically available knowledge sources as basis for learning the appropriate self-adaptation behaviour in SASO systems based on experiences?”

We call this concept ‘opportunistic meta-learning’, since it tries to make the best of the currently available resources for improving the run-time learning behaviour at a meta level [3]. This paper describes the motivation, the approach, and the current state of the PhD project. Next, we mention the Related Work on the field of meta-learning, clarify the used terminology (Section 6.2), and present the research questions that we want to tackle (Section 6.3). With this in mind, we present the architecture as a blueprint for an opportunistic knowledge adapting system (Section 6.4) and explain the required self-* properties (Section 6.5). As a first showcase, we present a concrete example scenario where the methodology can be applied to (Section 6.6). At the end, we summarise the discussed elements and give an outlook on further research (Section 6.7).

6.2 Related Work and Terminology

The field of Meta-Learning as a sub-domain of Machine Learning came up after classical Machine Learning had been understood and only the task of learning how to find the optimal model remained, which Calman et al. [3] denote as *learning to learn*. In the year 2002, Vilalta and Drissi distinguished Meta-Learning from classical Base-Learning by their levels of adaptation: Meta-Learning is able to adapt its bias dynamically, whereas it is constant in Base-Learning [18]. Later, Huaxiu et al. [21] introduced four state-of-the-art bias adaptation techniques: recurrent neural networks capable of storing Meta-Knowledge [7], learning a Meta-Optimiser [22], learning an effective distance metric between samples [14, 19], or learn an appropriate initialisation and adapt through few gradient steps [21]. Whereas Huaxiu et al. focus on the latter, which induces a dependency between the meta-adaptation and the base-learner since both, the initialisation and the gradient, has to be tuned for a specific base-learner, we decided to focus on the meta-optimiser. The term *opportunistic* in the context of a Machine Learning system was introduced by [2] and describes ‘a system that has to make the best of the available information or knowledge’. In order to establish knowledge exchange between agents, common protocols are required that

every agent understands. These protocols can be from the Internet domain, as Holger et. al. presented in [12], which used distance vector routing and link state routing. In detail, they showed that autonomously acting traffic light systems can guarantee improved route guidance in traffic by ensuring the synchronization of a green phase at points of high traffic volume.

By tightly following the definitions in [3], we give an overview of the most important phrases, i.e. *knowledge*, *knowledge source*, *naïve learner*, *meta learner*, and *opportunistic*.

Knowledge

As we do not want to constrict us to one certain representation of knowledge, like in Machine Learning where training sample instances are often used synonym to knowledge, we denote as knowledge or equivalently empirical data the manifestation of experience made by one or more entities called knowledge sources. The latter includes not only the classical training samples, which might be assigned with a label but also trained machine learning models themselves including their specific parametrisation and even the decision why a certain learning model was selected.

Knowledge Source

Knowledge source is an entity that provides knowledge. More specific, A. Calma et al. [3] made out a list of various knowledge sources:

- Human
- Other productive systems
- Context and environment
- Free, open data
- Collateral knowledge

Occasionally, one of these knowledge sources is available to be explored and exploited at runtime. This challenge is visualised in Figure 6.1.

Human

Humans can be experts that are able to fulfil the task of the autonomous system manually on their own, they also may be able to describe the cause of a novel sensor status. For these reasons, human knowledge is a valuable knowledge source but on the other hand a survey of human experts is often accompanied by high economic costs and can often not be used as an on-call service. D. Gorecky and M. Schmitt et al. presented a Human-machine decision loop where the human intervention is placed at last [5]. This has two direct benefits. First, machines can filter data and provide only decision-relevant data for the user. Second, even if machines were involved in the decision making process, the decision can be considered human which means the decision is easier to understand and accept.

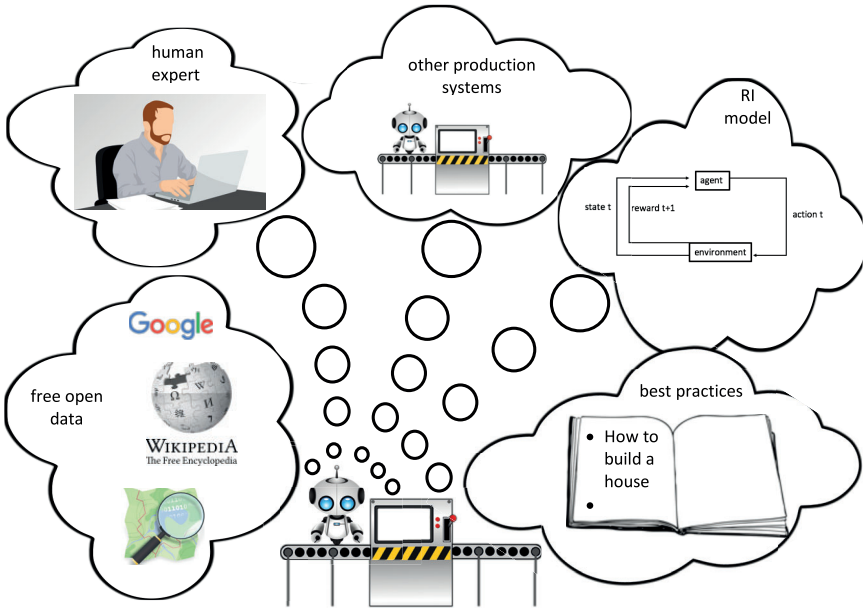


Fig. 6.1. Occasionally available knowledge sources.

Other productive systems

In contrast to the human-machine interactions, other productive systems may be used as knowledge source. Instead of hoarding knowledge in a monolithic place, several agents may store Knowledge that is interesting only for a specific problem or location. Whenever another agent finds the knowledge interesting it may ask for it.

Context and environment

If physical models of the environment are available, the agent may simulate certain possible actions. On the other hand, if a certain model is not provided, with the help of reinforcement learning agents are enabled to explore their environment without neglecting their target functions.

Free, open data

Some data can be queried from the web or other databases, i.e. weather data or stock market data.

Collateral knowledge

For certain situations there exist knowledge about the perfect action, also known as *best practices* as represented in Figure 6.1. One of the challenges here is the non-uniform sensor or action set. However, since best practices are often formulated in

general terms, it is often possible to assign the sensor state of an agent to the general state as described by the best practice.

Between these knowledge sources, we can clearly observe differences in various dimensions. Not only the quality and the availability are different but also the way of transference. One subset of the knowledge sources poses the knowledge in an interpreted state, and access to this knowledge is granted over communication (humans, other productive systems), whereas other knowledge sources (context and environment) need to be measured or collected.

Naïve Learner

A naïve learner is a classical Machine Learning model that expects training data either sample-wise or as a batch of data. After receiving the data, the model fits the data by approximating the hidden probability distribution. At the end, the performance of the trained model can be analysed by providing a test set of samples which are unknown for the model. The performance is highly sensible to a set of external model parameters, called *hyper parameters* which has to be adjusted before the training procedure. Here, we choose the term naïve to indicate that the model offers no possibility to select training data or to change its hyper parameters.

Meta Learner

The first term *meta* means above or outside, which means that the meta learner has a different perspective in comparison to the naïve learner. The meta learner reflects the behaviour of the naïve learner and compares the knowledge with knowledge from the environment. As this behaviour can be summarised to a system learns to learn, we call this system according to [3] a meta learner.

Opportunistic

The term opportunistic in the context of a learning system was used in [1] in order to express the greediness of a system to acquire all available knowledge sources at run-time.

6.3 Research Questions

We are motivated by the question how systems can self-reflect their knowledge in comparison to other knowledge in the environment while simultaneously providing the system's function. Therefore, we develop approaches for a continuous adaptation of the knowledge of an intelligent system in an opportunistic way, which means exploiting available knowledge at run-time. The latter requires an appropriate model to capture the state of knowledge sources in order to rank the knowledge sources according to an intelligent system's current needs. Knowing which knowledge sources is relevant given a temporarily available pool of such instances, leads to the question

under which protocol the knowledge transfer could be organised. Finally, in order to manage the knowledge sources correctly, their status should be monitored over time and an update routine should be developed that activates or deactivates knowledge sources. In order to solve this, the following research questions can be summarised:

- How to define a model of knowledge sources that enables learning systems to take advantage of occasionally appearing knowledge?
- Which protocol should be used to enable communication between the intelligent system and the knowledge source?
- How to manage knowledge sources by an intelligent system?
- How to implement an update routine to activate or deactivate knowledge sources at runtime?

6.4 Opportunistic Learning Architecture

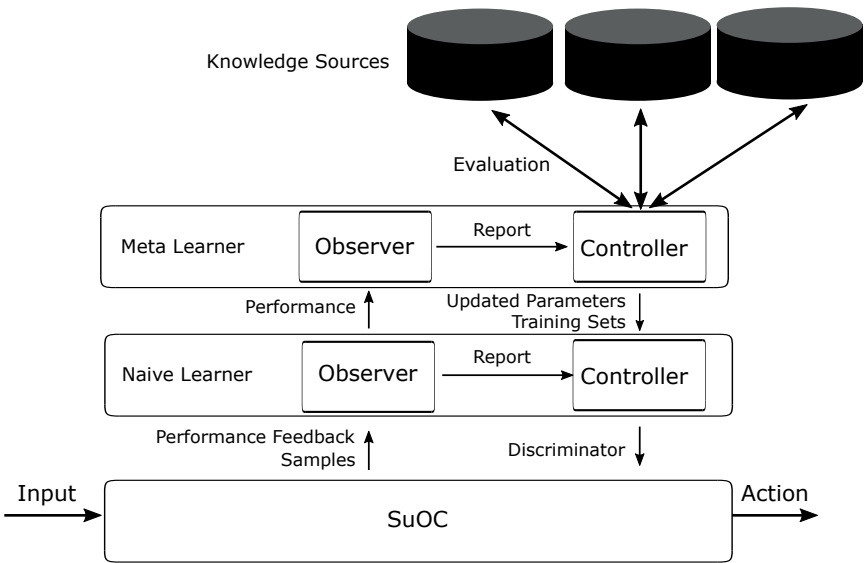


Fig. 6.2. Knowledge self-adaptation Observer/Controller framework.

In the traditional Observer/Controller framework, there are only two types of layers. Layer 0 is called System under Observation and Control (SuOC) and the upper layers are called Control Mechanisms (CM). To explain the concept of knowledge adaptation, we rename the layers to clarify that the concept is a specific version of the Observer/Controller framework. Hence, we stack up three layers upon each other as seen in the Figure 6.2, namely, the System under Control and Observation

at the bottom, the naïve learner in the middle, and the meta learner on top. The layers of the knowledge adaptation framework are created bottom-up but monitoring and controlling is done top-down. That means, the layers do not rely on their upper layers but the upper layers contribute to the system as a whole. Instead of using a bottom-up escalation mechanism, which would add another responsibility to the layer, the observer in the Observer/Controller control mechanism one layer above monitors the behaviour of the layer beneath. This results in a stack of systems that are only monitoring downwards and are still capable of working without their managing superior layers.

In order to describe the learning procedure of the different entities, we assume a scenario without *a priori* knowledge, that means the first observation at the input is the first knowledge the system acquires. Depending on the kind of input which arrives at the SuOC, we demonstrate three typical cycles: First, in case the sample is very frequent, the SuOC might choose the correct action according to a default discriminator, i.e. choose the same action for all incoming observations. Whether the action was correct or not is provided for some of the observations and can be implemented by a test loss function (cf. *Performance Feedback* in Figure 6.2). Second, if the observation is more rare, the default action might lead to a negative *Performance Feedback*. The observer of the naïve learner reports these news to the controller which may decide to exchange the discriminator of the SuOC with the decision function of a fresh trained standard ML model. This model is trained on samples which has been observed over time. Third, for a sample that appears to the naïve learner as new either in the sensor characteristic or in the necessary action, the meta learner observes a negative performance and provides the naïve learner with updated parameters and training sets that includes the novel sample. These training sets might be explored via offline learning and simulations or might be taken from one of the knowledge sources. Periodically, the meta learner updates a list of knowledge sources that are ranked according to an information gain function that is weighted by the class distribution of the input stream (knowledge sources containing more likely classes will be preferred) and the class-loss of the naïve learner (knowledge sources containing more classes the naïve learner misclassifies are preferred).

6.5 Self-* Properties

In the light of IBM's vision for Autonomic Computing [6] and the subsequent vision of Organic Computing [17], an autonomous system should fulfil the key elements of an Autonomic Computing system. Some of them can be defined as self-* properties, like self-optimisation, self-healing or self-protection. In the following, we list the most important self-* properties required for our concepts.

6.5.1 Self-Optimisation

Self-loops that keep a system's state within a well-defined working state are well known from control theory [10]. However, we think in the case of knowledge ad-

aptation, permanently applying a feedback control loop after every iteration is too greedy.

6.5.2 Self-Healing

A required self-healing technique might be the replacement of a whole learning model in case of malfunction. Multiple backup models may be trained for this purpose.

6.5.3 Self-Protection

Especially for an opportunistic learning system it is important to make sure that the opportunistic behaviour is not exploited by an attacker, who might also be able to apply adversarial attacks. Fooling the system with a malicious data set can be harmful and is not easily detectable for humans. For example in [4], images with a minimal amount of noise are misclassified by neural networks, whereas humans are not aware of the corruption.

6.5.4 Self-Adaptation

Implementing a self-adaptation system according to Salehie and Tahvildari [13] we have to answer the 5W+1H questions specifying the adaptation further. Simply listed, the questions are: *where*, *when*, *what*, *why*, *who*, and *how*. However, Krupitzer et al. [9] mentioned that the question about who is adapting is trivial as the adaptation is done by the self-adaptation system and not by humans. For the presented architecture we categorise two different types of adaptation: We denote the first adaptation between layer 2 and 1 as the meta layer or also naïve learner adaptation. That is, the meta layer can adapt the naïve learner either by changing the hyper parameters of the model or replacing the learner with a different classical machine learning model. The second adaptation between the naïve learner and the SUOC noted correspondingly naïve learner/SuOC adaptation and controls the SuOC by the scenario-specific control parameters.

Where?

In order to correspond to the Observer/Controller framework [16], the adaptation should always be applied top-down from a superior to the lower layer and should affect only the one layer underneath. This methodology carries two advantages. First, the layer can directly observe the effects of its adaptation since it provides an observer to the layer beneath. Second, for each layer, there exist at maximum one adaptor. This means, there are no race conditions and the adaptations can be concerted synchronously.

When?

As the control mechanism is fully responsible for its underlying layer, it decides when the adaptation happens. Internally, this can be implemented by an escalation function that measures the performance and escalates if a certain threshold is reached. Depending on where the adaptation took place, the process acts on different time scales. Because the naïve learner observes the SuOC itself, in case of significant malfunction we want to adapt the SuOC as fast as possible. On the other hand, an analysis of whether and when to adapt the naïve learner is on a larger time scale and a replacement of the naïve learner or its knowledge source requires long-time simulation runs to proof the performance of the new learner. Here, the training set, the type of Machine Learning model, and its hyper parameter has to be adjusted. However, offline learning procedures are able to train backup models and find optimal hyper parameters independently from the time constraints of the SuOC. These backup models might also be found in other Knowledge Sources.

What?

Every kind of knowledge samples, model parametrisations, and machine learning models should be adaptable.

Why?

The key strength of the system's adaptation is the ability to find an adequate transformation even to the most unlikely unpredictable events like the arrival of inputs of new classes or a completely unbalanced class distribution.

How?

The adaptation process should be as smooth as possible, so that a continuous functioning of the productive system is guaranteed. Especially for the model adaptation, a promising approach is to let both systems, the old and the optimised one, run in parallel for a short adaptation period T . During this period at time t the old model's decision is taken into account with a probability $p(t) = 1 - \frac{t}{T}$ and the decision of the new model is considered in $1 - p(t)$ of the cases.

6.6 Scenario: Quality Assessment at a Conveyor Belt

We use quality assessment as an example scenario, where the task is to analyse the quality of produced workpieces to decide whether the products can be sold or not. This includes different error classes for different quality levels. In our approach, the basic control loop is realised by means of observation, control, and productive parts (i.e., the surface inspection system as a SuOC).

The SuOC in Figure 6.3 is just assigning the workpieces to different buckets (for delivery or failure classes), while the decision is guided by the adaptation loop. The

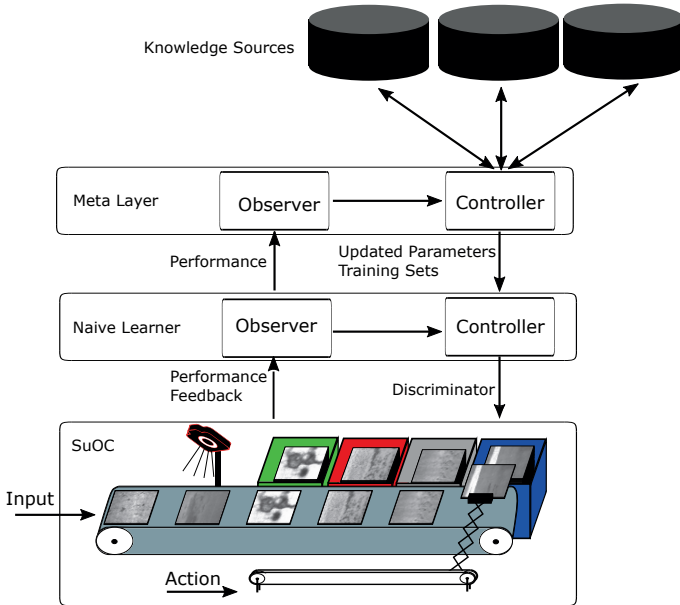


Fig. 6.3. Quality assessment as a scenario for opportunistic self-awareness of meta-learning components.

system starts without knowledge and has to make use of the dynamic sources (such as humans, best practises, or other systems) depending on their availability, certainty, or cost. In the simulation of the conveyor belt, we are streaming images of a hot-rolled steel strip to the SuOC. We use the Northeastern University (NEU) surface defect data set [15] with 6 error types (examples are shown in Figure 6.4) providing 300 images for each.

To these 1800 images we add the same amount of defect-free clones and extend the data set by a multiple via augmentation. In the images, one out of six error types can be found, which are exemplary listed in Figure 6.4. In detail the error types are *crazing (Cr)*, *inclusion (In)*, *patches (Pa)*, *pitted surface (PS)*, *rolled-in scale (RS)*, and *scratches (Sc)*. After receiving the sample, the SuOC classifies it according to a static discriminator. The classification may subsequently trigger a feedback responding with the ground truth label with a certain probability.

These information are dynamically observed by the naïve learner above which compares the predictions with the truth values and calculates a performance value. In our application, we used a history-based accuracy. When the performance falls below a certain threshold, the samples and truth labels are used to train a static machine learning model. After training, the old discriminator is replaced by the new model if its performance is better.

On a higher level, the meta learner trains multiple machine learning models, like decision trees, support vector machines, and neural networks. Additionally, it extends

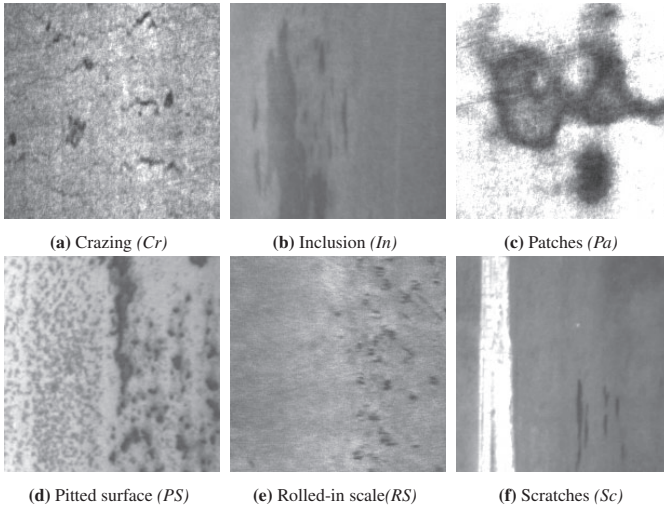


Fig. 6.4. Northeastern University (NEU) surface defect database [15].

also the observed knowledge with knowledge from the top k knowledge sources ranked by their Kullback–Leibler divergence of their sample and class distribution. If the performance difference of the trained meta learner model and the naïve learner model exceeds a fixed threshold, the latter is replaced.

Furthermore, we model knowledge sources of different quality. The knowledge sources contains classified images of the NEU surface defect data set which are not used for the streaming. From these, we generated synthetic data by focusing on the introduction of three types of disturbances or defects:

- *unreliable*
- *distorted*
- *novel*

In *unreliable* KS, a percentage of the samples is misclassified. *Distorted* KS is based on images with an additive, zero-mean Gaussian noise of standard deviation σ . Finally, in *novel* knowledge sources, there are samples that have not been observed up to the time step t_i .

Finally, the performance in terms of accuracy is monitored while stressing the system with noisy images, new classes, or unreliable knowledge sources. In Figure 6.5 the advantage of the knowledge source adapting system can be seen, which is able to increase the class accuracy of the defect class *RS*.

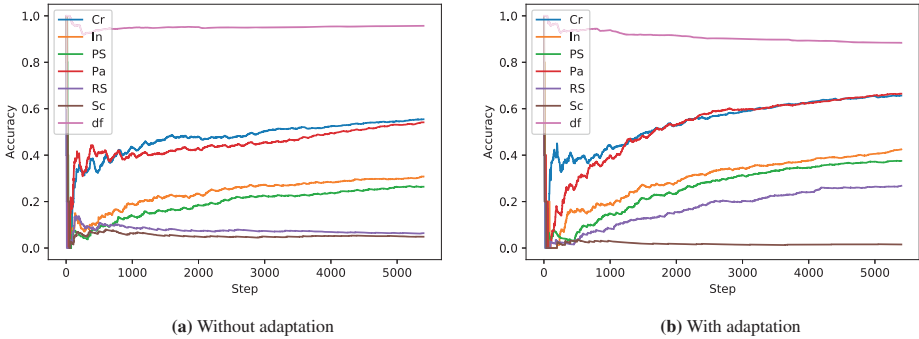


Fig. 6.5. Class accuracy comparison of Multi-Layer Perceptron with (*left*) and *without* knowledge source adaption.

6.7 Summary and outlook

The paper has shown the concepts of a PhD project in the domain of self-adaptation systems. After clarifying terminology terms, a concrete O/C architecture with a specification is presented. Subsequently, the knowledge self-adaptation abilities of this architecture are analysed by answering the 5W+1H questions. In the end, a scenario is presented which demonstrates a proof of work in the domain of quality assessment.

Given the presented scenario in quality assessment, a possible layer on top of the O/C layers is imaginable which aggregates the whole knowledge learned so far and is able to exchange this knowledge package and asks for more packages from other learning systems that provide the same communication layer. Besides the simulation of the conveyor belt, other scenarios are planned to be investigated. For example, finding the shortest path to a target within a randomly generated maze is worth to be analysed because of the limited sensor view and the local restrictions of agents distributed at random positions in the maze. As the imperfect knowledge is another difficulty in this scenario, the agents might then communicate over a channel in order to help each other finding the target as fast as possible. As random maze generation is computationally cheap, the usage of knowledge of previous runs as starting point can be easily evaluated. Other scenarios are planned to show certain aspects of the knowledge self-adaptation system. We think that the future of learning especially in the domain of predictive maintenance cannot be handled with static machine learning models engineered at design-time as there are always unpredictable observations to which the system should react to at run-time.

References

1. Bahle, G., Calma, A., Leimeister, J.M., Lukowicz, P., Oeste-Reiß, S., Reitmaier, T., Schmidt, A., Sick, B., Stumme, G., Zweig, K.A.: Lifelong learning and collaboration of smart technical systems in open-ended environments – opportunistic collaborative interactive learning. In: 2016 IEEE International Conference on Autonomic Computing (ICAC). pp. 315–324 (2016)
2. Bahle, G., Calma, A., Leimeister, J.M., Lukowicz, P., Oeste-Reiß, S., Reitmaier, T., Schmidt, A., Sick, B., Stumme, G., Zweig, K.A.: Lifelong learning and collaboration of smart technical systems in open-ended environments–opportunistic collaborative interactive learning. In: 2016 IEEE International Conference on Autonomic Computing (ICAC). pp. 315–324. IEEE (2016)
3. Calma, A., Kottke, D., Sick, B., Tomforde, S.: Learning to learn: Dynamic runtime exploitation of various knowledge sources and machine learning paradigms. In: 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W). pp. 109–116. IEEE (2017)
4. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
5. Gorecky, D., Schmitt, M., Loskyll, M., Zühlke, D.: Human-machine-interaction in the industry 4.0 era. In: 2014 12th IEEE International Conference on Industrial Informatics (INDIN). pp. 289–294 (2014)
6. Horn, P.: Autonomic Computing: IBM’s Perspective on the State of Information Technology. Tech. rep., IBM (2001)
7. Hospedales, T., Antoniou, A., Micaelli, P., Storkey, A.: Meta-learning in neural networks: A survey. arXiv preprint arXiv:2004.05439 (2020)
8. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* 36(1), 41–50 (2003)
9. Krupitzer, C., Roth, F.M., VanSyckel, S., Schiele, G., Becker, C.: A Survey on Engineering Approaches for Self-adaptive Systems . *Pervasive and Mobile Computing* 17(B), 184 – 206 (2015), 10 years of Pervasive Computing’ In Honor of Chatschik Bisdikian
10. Lee, E.B., Markus, L.: Foundations of optimal control theory. Tech. rep., Minnesota Univ Minneapolis Center For Control Sciences (1967)
11. Müller-Schloer, C., Tomforde, S.: Organic Computing - Technical Systems for Survival in the Real World. Birkhäuser (2017), <https://doi.org/10.1007/978-3-319-68477-2>
12. Prothmann, H., Tomforde, S., Lyda, J., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Self-organised routing for road networks. In: Kuipers, F.A., Heegaard, P.E. (eds.) *Self-Organizing Systems*. pp. 48–59. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
13. Salehie, M., Tahvildari, L.: Self-adaptive software: Landscape and research challenges. *TAAS* 4 (01 2009)
14. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: *Advances in neural information processing systems*. pp. 4077–4087 (2017)
15. Song, K., Yan, Y.: A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science* 285, 858 – 864 (2013), <http://www.sciencedirect.com/science/article/pii/S0169433213016437>
16. Tomforde, S., Goller, M.: To adapt or not to adapt: A quantification technique for measuring an expected degree of self-adaptation. *Comput.* 9(1), 21 (2020)
17. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic Computing in the Spotlight. arXiv.org (January 2017), <http://arxiv.org/abs/1701.08125>

18. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. *Artificial Intelligence Review* 18(2), 77–95 (2002)
19. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: *Advances in neural information processing systems*. pp. 3630–3638 (2016)
20. Weiser, M.: Hot topics-ubiquitous computing. *Computer* 26(10), 71–72 (1993)
21. Yao, H., Wei, Y., Huang, J., Li, Z.: Hierarchically structured meta-learning. *arXiv preprint arXiv:1905.05301* (2019)
22. Yapici, H., Cetinkaya, N.: A new meta-heuristic optimizer: Pathfinder algorithm. *Applied Soft Computing* 78, 545–568 (2019)

Verification of Open Stochastic Heterogeneous Systems with Stochastic Contracts

Jens Sager

OFFIS - Institute for Information Technology, 26121 Oldenburg, Germany, jsager@offis.de

Abstract. The transition from fossil fuels to renewable energy sources is characterised by an increased decentralisation of power generation. Classical control approaches are changed in favour of decentral solutions, promising better scalability and adaptability. The resulting increase in complexity may result in worse grid stability. To alleviate this issue this work seeks to create a verification system for open, heterogeneous, stochastic systems by extending existing stochastic contract approaches with reasoning about a changing number of components. We explore the addition of operators to Stochastic Signal Temporal Logic (StSTL) contracts that establish relationships between variables on different system levels, encompassing all components of a given type. This allows us to quantify properties when the component count changes during runtime.

Keywords: Contracts, Verification, Open Systems, Rely/Guarantee Reasoning.

7.1 Introduction

The transition towards renewable energy sources is characterised by the use of large numbers of small generators. Additionally, market participation becomes increasingly more decentralised, allowing access to smaller producers. An individual unit's generation also becomes less dependable with photovoltaic (PV) and wind generators being heavily dependant on the weather. This means a large number of different participants may or may not be able to contribute to ancillary services at a given time. When moving from large scale fossil power plants with predictable outputs to these new renewable sources, classical approaches to stabilise the energy grid have to be adapted.

This has resulted in a push to apply known schemes of decentralised control to the new energy grid landscape. By applying principles of Organic Computing [14], these systems can exhibit desirable self- $\{$ adaptive, healing, governing, $\dots\}$ properties emerging from simple component behaviours. The cost is an increase in complexity and reduced understanding of the system as a whole [21]. Consider the approach presented by Lehnhoff et al. in [10]. They propose providing frequency response reserve (FRR) with distributed energy resources (i.e. generators and batteries in

different physical locations) by creating coalitions of agents. FRR is a predetermined amount of power that must be made available by a provider in case of a supply/demand imbalance, visible through a change in grid frequency. This power must be fully available within 30s and kept available for at least 15min. In the presented approach, generation units are matched into different pools for providing FRR based on their opportunity costs, reliabilities and the small-signal stability (i.e. stability under small disturbances) of a coalition.

The energy grid is a critical infrastructure and thus needs a special level of certainty in any new technology introduced to it. The above approach is simulated but not verified. We seek to alleviate that gap by providing a framework to verify properties of systems made up of a varying number of different components, each with its own stochastic behaviour. To achieve this, we first identify an initial contract framework and extend it with operators to allow reasoning over a variable number of components. We base this work on a system for providing a simplified version of FRR with distributed batteries.

The remainder of this paper is structured as follows. Section 7.2 gives an overview of the state of the art in verifying decentralised systems. In Section 7.3 the research gap is identified and the main goals of the dissertation as well as the research questions are explained. In Section 7.4 we show the criteria for our contract logic, the evaluated logics and the decision process. Section 7.5 shows a first contract for a system providing FRR to the energy grid and identifies operators to allow reasoning over variable component counts. Finally, in Section 7.6 we present the next steps of the project and give a conclusion in Section 7.7.

7.2 State of the Art

This section provides an overview of the state of the art for the verification of decentralised systems. The investigated starting points are the verification of swarms of autonomous robots and the use of assume/guarantee (alternatively rely/guarantee) reasoning to generalise over types of system components. Contracts are identified as a popular variant of the assume/guarantee approach in the development of Cyber-physical systems (CPS). In particular, we are interested in the static verification of systems (i.e. without running the system).

7.2.1 Verification for Robot Swarm Behaviour

A robot swarm is a collection of simple autonomous robots, designed to cooperatively achieve a common goal. Inspired by swarming behaviour in nature, their individual abilities are usually very limited. The control challenge is to use simple rules and large numbers of robots to achieve the goal as an emergent property. This is similar to the goal of achieving certain energy grid properties through distributed autonomous participants. Thus, robot swarm verification addresses similar challenges as faced in the energy context.

Kouvaros and Lomuscio have done extensive research into verifying properties for robot swarms. Their focus is on swarms of equal agents, where each agent has an internal state, a set of actions it can perform and deterministic [8] or probabilistic [13] way of choosing an action. Their main verification approach is assuming some fixed number of agents and proving a desired property for this system. Since agents can be added or removed from the swarm this is not sufficient. To generalise, they look for a cut-off theorem [7] meaning a certain number of agents exists such that for every larger swarm the property holds. This is also reflected in their definition of emergence. A property is emergent if there exists an emergence threshold n such that a system with at least n agents fulfils the property [12] as shown in Figure 7.1.

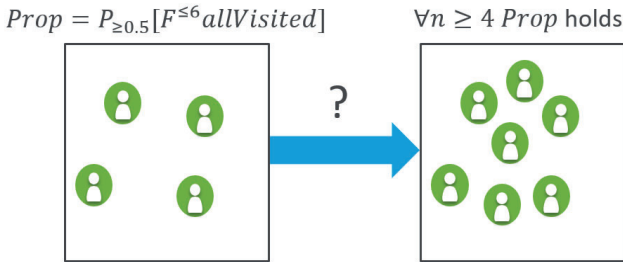


Fig. 7.1. Concept of an emergence threshold as introduced in [12]: A property always holds when the swarm consists of at least n robots.

For deterministic agents this allows them to verify properties of the form $\phi = \forall_{\{v\}} K_v GF con_v (ACTL^* K_{-X})$ [7], meaning each robot v knows (K_v) it will be connected (con_v) infinitely often (GF). These properties are formulated in $ACTL^* K_{-X}$. That is the superset CTL^* of Computational Tree Logic (CTL) and Linear Temporal Logic (LTL) extended with epistemic properties (reasoning about knowledge, K) and without the X (next) operator of CTL^* . This formalism is further extended with indexed atomic propositions. This is used to distinguish between parametrised systems over an unbounded number of agents and concrete systems over a specific number of agents. It is used to verify fault tolerance in parametrised multi-agent systems with an unknown number of agents during the design phase. [9]

For probabilistic agents they define probabilistic LTL ($PLTL$) properties of the form $P_{\geq 0.5}[X^6(connected, 1)]$. This means, the agent with ID 1 will be connected after 6 time steps with probability of at least 0.5 [13]. For the subset of bounded time properties they provide sound and complete algorithms for emergent property identification (EPI) and emergence threshold identification (ETI) [12]. Extending the expressiveness to broader properties and the question of finding a minimal emergence threshold remain future work.

Cavalcanti et al. [1] have extended RoboChart—a domain specific language for the modelling and verification of single robot systems—with facilities to verify heterogeneous swarms of robots. They achieve this by introducing collections that specify

how many robots of each (predefined) type are present in the swarm. Communicating Sequential Processes (CSP) are derived for each module. Additional aggregation processes to capture system level properties are introduced and the entire model is checked with the theorem prover Isabelle. The explicit definition of global properties as well as probabilistic models and environment specifications are planned extensions to the language.

Leofante et al. [11] propose a hybrid systems approach to swarm robotics, modeling the robots with hybrid automata and showing swarm properties via reachability analysis. Verification is done for swarms of fixed size. They test their approach for two case studies: synchronizing LED flashing of robots and collision avoidance while exploring a square environment.

Konur et al. [6] present a probabilistic model checking approach to verify swarms of identical robots. A robot is modelled as a probabilistic finite state machine. System properties are defined in PCTL and analyzed with the PRISM model checker. Large numbers of robots can be handled by using a counting abstraction technique (similar to [8]). They introduce probabilistic methods to consider general population sizes in a given system state, rather than calculating the actual number of robots. The approach is tested with a foraging robot scenario but can be generalised to any robot that can be modelled as a probabilistic finite automaton.

7.2.2 Decomposition and Rely/Guarantee Reasoning

Rely/guarantee reasoning specifies the function of a system component by giving a set of properties it can expect from its environment (relies, sometimes also called assumptions) and a set of properties it guarantees to other components (guarantees). This is an effective method of abstracting implementation detail in a specification.

Nafz et al. [15] verify self-organizing systems with an observer and a controller (Observer/Controller architecture) as shown in Figure 7.2. The normal function of the system is defined by an invariant. As long as the invariant holds the controller remains inactive. If the invariant is violated the functional system moves into a quiescent state and is reconfigured by the controller back to working condition. Thus the approach is named the Restore Invariant Approach (RIA).

Their verification is based on decomposing the system into the functional part, consisting of a number of independent agents and an Observer/Controller part. Each system component has relies and guarantees. Their main result is a composition theorem in Rely Guarantee Interval Temporal Logic (RGITL), an extension of Interval Temporal Logic (ITL) with relies and guarantees. By this theorem, a system-wide property may be verified by verifying a set of component level properties. Thus a system property may be decomposed to properties of the individual agents. If the agents have a limited number of different types, then each agent type only needs to be verified once.

A different approach using rely/guarantee reasoning is contract-based design [19]. Here, system components are also modelled by sets of assumptions and guarantees. Contracts may be defined both on the same system layer (horizontally) and across layers (vertically). Thus vertical contracts can be used to abstract implementation

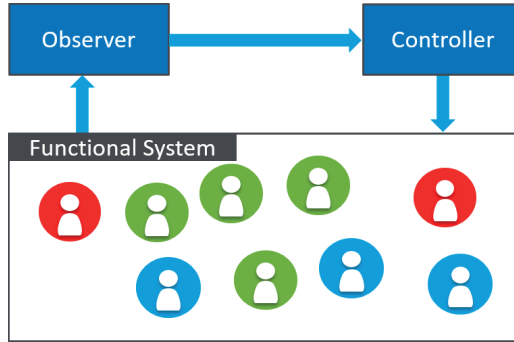


Fig. 7.2. Observer/Controller Architecture resembling the basic concept presented in [23] and [22].

detail of lower system levels. They define contract composition and contract refinement via set operations on the assumption and guarantee sets of different contracts. The composition of two contracts C_1 and C_2 combines them into a single contract that holds when both C_1 and C_2 hold. C_1 refines C_2 if it has stricter guarantees and relaxed assumptions. Thus whenever C_1 is satisfied C_2 must be as well. The verification process can then be broken down into computing the composition of all component contracts and showing that the composite contract is a refinement of the system contract.

An extension to probabilistic contracts with properties encoded in Stochastic Signal Temporal Logic (StSTL) formulas is presented by Nuzzo et al. [16]. They provide tools for the composition and refinement of their contracts by encoding bounded StSTL formulas into Mixed Integer Programs (MIP) and using established solvers for contract verification and control synthesis. Because their assumptions and guarantees are not given as sets but as StSTL formulas, their contract operations are also defined with regards to these formulas.

7.3 Research Gap

For the verification of electrical grid properties with decentralised components that may enter or leave the system autonomously, the following properties are desirable:

- heterogeneous - support for different types of components
- stochastic - outcomes are given with a probability
- open - components may enter or leave the system at runtime

Table 7.1 shows which of the desirable properties are fulfilled by the state of the art approaches. None contains all three. The identified research gap therefore is the construction of a framework for the modelling and verification of open stochastic heterogeneous systems. The ultimate goal is to create a solver that — given a set of component types and a system goal — can compute a combination of components

Table 7.1. Overview of State of the Art in relation to desirable framework properties.

	State of the Art						
	[9]	[1]	[11]	[6]	[15]	[19]	[16]
Properties open	×	×		×	×		
stochastic	×			×			×
heterogeneous		×	×		×	×	×

that will fulfil the system goal. Types and goals will be specified by assumptions and guarantees. This provides an intuitive decoupling of the expected system behaviour and its implementation. The main challenges is showing how contract refinement and contract composition can be generalised to changing numbers of stochastic components.

7.3.1 Goals

In the following, we illustrate the two main goals of this work. The first is system verification, shown in Figure 7.3. Given a system, specified by a system contract consisting of assumptions (A) and guarantees (G), a set of known component types, and the possible amounts of each component in the system (**component space**), verify that the contract is satisfied for all possible combinations of components. Additionally, this should include that contract satisfaction is stable under changing component numbers, i.e. under transitions in the component space.



Fig. 7.3. System Verification: Given a system contract (A,G), a set of component types, and their allowed amounts in the system, prove that any possible combination of components fulfils the contract.

The second goal, system synthesis, is illustrated in Figure 7.4. Given a set of known component types and a system contract, generate combinations of components that will satisfy the system contract. This can be an explicit solution, i.e. fixed numbers

for each type, or a general solution describing the ratios of components necessary. The main difference to the verification goal is that the component space is not necessarily restricted and a single solution in the component space is sufficient (transitions are of no concern). In practice, restrictions to the component space will probably speed up finding a solution.

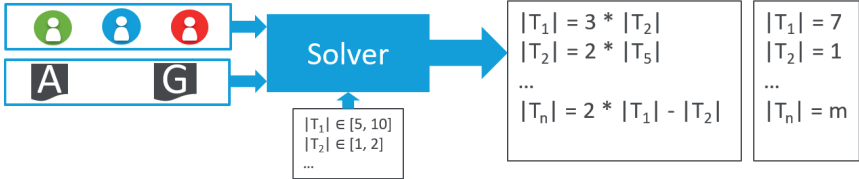


Fig. 7.4. System Synthesis: Given a system contract (A,G) and a set of component types, generate a combination of components that fulfil the contract. Optionally, an allowed range of components may be provided.

7.3.2 Research Questions

To achieve these overarching goals, the following research questions should be answered in the PhD project.

RQ1 How can open, stochastic, heterogeneous systems be verified using contract specifications.

- a) How can contract operations be extended to be applicable to open systems?
- b) How can satisfiability of such a system contract be verified?
- c) Given a space of components and a system contract, how can we determine if all component combinations in the space satisfy the system contract?
- d) How can stability of contract satisfaction under transitions in the component space be verified?
- e) What characteristics are necessary for a system with a contract that is stable under component space transitions?

RQ2 How can component combinations for open, stochastic, heterogeneous systems be synthesised from a contract specifications.

- a) How can the existence of a solution be determined?
- b) How can explicit solutions be generated?
- c) How and under what conditions can general solutions be generated?

7.4 Logic Selection

In this section we summarise the selection process for an initial logic to focus on in this work. We evaluate a set of temporal logics against criteria derived from an example system in the energy domain.

7.4.1 Properties of a Contract Logic

A fundamental question going forward is with which logical framework these new contracts can be achieved. An initial idea would be to extend any of the state of the art formalisms with the properties they are lacking. These are, however, widely different temporal logics with properties that would have to be proven again with any changes made to them. Konur [5] conducted a survey and classification on temporal logics. In general, these can be distinguished by their logic order (propositional or first-order), their fundamental entity (time point or time interval) and their temporal structure (linear or branching).

To evaluate the logics, a generalised system for providing FRR is modelled with each. The system contract should encode the following properties:

- A frequency violation is defined as: $vio := |f - 50| > 0.2$ where f is the grid frequency in Hz.
- The predetermined reserve power P_{target} is available to the grid within 30s of a frequency violation occurring.
- Once the target power is reached it is held for at least 15min.
- The system contract holds with a probability of at least 95%
- The system components may then consist of any set of generators, consumers and batteries that independently can observe the frequency and fulfil a component contract.

Note that in reality the power output is scaled with the size of the frequency violation and must be provided as both positive and negative power depending on the direction of the violation. We neglect this for simplicity and provide the full positive power at all times when a violation occurs. Based on this, the logics are evaluated on the following criteria:

1. Can the system contract be modelled accurately?
2. Do necessary contract operations already exist or do we have to develop them?
3. Does an extension to open stochastic systems and contract reasoning break important logic properties?
4. Is the logic easy to use for defining the system contract?
5. Is the notation intuitive?

7.4.2 Evaluation and Decision

The evaluated logics are PLTL [2] (a probabilistic extension of LTL), the probabilistic neighbourhood logic PNL [3], Simple Probabilistic DC (SPDC) [24], Interval

Duration Logic (IDL) [17], the Stochastic Signal Temporal Logic (StSTL) [16] and Probabilistic Signal Temporal Logic (PrSTL) [18], the probabilistic extension of Computation Tree Logic (PCTL) [4] and finally the extension of ITL with rely/guarantee reasoning RGITL [20]. For each criterion the logics are given one of these scores:

- ++ criterion applies fully
- + criterion applies mostly
- o criterion applies partially or can not be properly evaluated
- criterion does not apply

The first three criteria can be objectively evaluated. The fourth and fifth criterion are more subjective and evaluate if the system properties can be easily defined with existing parts of a logic and how easily the underlying notions of a logic are applied to the energy context. For example, PNL can encode time bounds of an interval by defining properties of its neighbouring intervals (hence the name). This is somewhat unwieldy when modelling the example system so the ease of use was evaluated as o. It also was very different intuitively from the other logics. The final results of the analysis are shown in Table 7.2.

Table 7.2. Results of the temporal logics analysis for applicability in the PhD project

Logic	accurate	contracts	extension	easy	intuitive
PLTL	o	-	+	o	+
PNL	++	-	o	o	-
SPDC	++	-	-	o	+
IDL	++	-	o	++	+
StSTL	++	++	o	++	+
PrSTL	++	-	o	+	o
PCTL	++	-	-	++	+
RGITL	o	++	o	-	+

The main contenders for selection were PCTL, RGITL and StSTL. RGITL has rely/guarantee reasoning and an existing decomposition theorem that is close to the contract operations we want to implement. This theorem makes heavy restrictions on the relies between system and component level, however and the overall modelling process was tedious compared to the other two. PCTL was very pleasant to use and could express the required properties but lacks any contract operations. It is unclear, how the logic could be expanded to include an undetermined number of components. This leads to the final decision of using StSTL because it has existing contract operations and solvers that can be used as a guideline for the extended logic. As a small caveat it is not entirely clear if modifications to use varying numbers of components will break some of these desirable properties. We will tackle these issues as we encounter them.

7.5 Modelling a Frequency Response Reserve System in StSTL

This section introduces an example system for providing FRR modelled in StSTL. We use it to motivate some of the extensions we will introduce to adapt the logic to varying component numbers.

7.5.1 Introduction to StSTL

StSTL formulas are defined for stochastic systems. The following are the definitions as used in [16]. A **probability space** is a triple $(\Omega, \mathcal{F}, \mathcal{P})$ where Ω is a set of outcomes, \mathcal{F} is a set of events and $\mathcal{P} : \mathcal{F} \rightarrow [0, 1]$ is the function assigning probabilities to events. \mathcal{F} is assumed to be a **σ -algebra**. That means \mathcal{F} is a non-empty set of subsets of Ω that is closed under complement and countable union. Elements of a σ -algebra are called **measurable sets**. An ordered pair (T, \mathcal{T}) of a set T and a σ -algebra \mathcal{T} is called a **measurable space**. A function between measurable spaces is a **measurable function** if the pre-image of every measurable set is measurable. This means given two measurable spaces (S_1, Σ_1) and (S_2, Σ_2) , a function $f : S_1 \rightarrow S_2$ is measurable if for all $Y \in \Sigma_2$ $f^{-1}(Y) := \{x \in S_1 \mid f(x) \in Y\} \in \Sigma_1$. Given a **random variable** $v : \Omega \rightarrow \mathbb{R}^{n_v}$, \mathcal{F}_v denotes the σ -algebra generated by v , i.e. the smallest σ -algebra on Ω that makes v measurable. A **random process** $\mathbf{w} : \Omega \times \mathbb{N}_0 \rightarrow \mathbb{R}^{n_w}$ adds a **filtration** $\mathbb{F} = \{\mathcal{F}_k\}_{k \geq 0}$, that is an increasing sequence of σ -algebras to a probability space. Each element \mathcal{F}_k is a σ -algebra generated by the sequence $\{w_t\}_{t=0}^k$ with $\mathcal{F}_{-1} = \{\emptyset, \Omega\}$.

A **stochastic system** is a tuple $\mathbf{S} = (z, x_0, \mathbf{w}, f)$ with

- $z = (x, u, w)$ the set of **system variables** separated into state variables x , control input variables u and environment input variables w with cardinality n_x, n_u, n_w respectively,
- $x_0 \in \mathbb{R}^{n_x}$ the initial state,
- $\mathbf{w} : \Omega \times \mathbb{N}_0 \rightarrow \mathbb{R}^{n_w}$ a random process describing the environment behaviour,
- f an arbitrary measurable function

The state process $\mathbf{x} : \Omega \times \mathbb{N}_0 \rightarrow \mathbb{R}^{n_x}$, control input process $\mathbf{u} : \Omega \times \mathbb{N}_0 \rightarrow \mathbb{R}^{n_u}$ and environment process \mathbf{w}_k describe the system dynamics such that:

$$x_{k+1} = f(x_k, u_k, w_k)$$

In the following, we give the syntax and semantics of StSTL. StSTL formulas are built on atomic predicates (or chance constraints) of the form:

$$\mu^{[p]} := \mathcal{P}\{\mu(v) \leq 0\} \geq p$$

Here, μ is a real valued measurable function, v is a random variable on the probability space $(\Omega, \mathcal{F}, \mathcal{P})$ and $p \in [0, 1]$ is a probability. A chance constraint is true if and only if the inequality $\mu(v) \leq 0$ holds with probability of at least p . Additionally, a predicate may be deterministic, meaning it is true if and only if $\mu(v) \leq 0$ actually holds, regardless of p . In this case, the superscript $[p]$ may be omitted.

The syntax of StSTL is given by the following grammar where $\mu^{[p]}$ is a chance constraint and ψ and ϕ are StSTL formulas:

$$\psi := \top \mid \mu^{[p]} \mid \neg\psi \mid \psi \vee \phi \mid \psi U_{[t_1, t_2]} \phi$$

Other temporal and logical operators, such as conjunction (\wedge), eventually (F) and globally (G), can be derived from this syntax.

Given a system behaviour z and a time k , we say that an StSTL formula ϕ holds at time k of z , written as $(z, k) \models \phi$ if and only if ϕ holds for the remainder of the sequence starting at time k . We write $z \models \phi$ if $k = 0$. With this, the semantics of any StSTL formula is defined as follows (the definition of the globally operator is also listed because we use it later):

$$\begin{aligned} (z, k) \models \top &\leftrightarrow \text{true} \\ (z, k) \models \mu^{[p]} &\leftrightarrow \mathcal{P}\{\mu(z_k) \leq 0\} \geq p \\ (z, k) \models \neg\psi &\leftrightarrow \neg((z, k) \models \psi) \\ (z, k) \models \psi \vee \phi &\leftrightarrow (z, k) \models \psi \vee (z, k) \models \phi \\ (z, k) \models \psi U_{[t_1, t_2]} \phi &\leftrightarrow \exists i \in [k + t_1, k + t_2] : (z, i) \models \phi \wedge \\ &\quad (\forall j \in [k, i - 1] : (z, j) \models \psi) \\ (z, k) \models G_{[t_1, t_2]} \psi &\leftrightarrow \forall i \in [k + t_1, k + t_2] : (z, i) \models \psi \end{aligned}$$

7.5.2 System Contract

In this section we model the system contract for our example FRR system. This means defining the necessary atomic predicates and combining them into assumption (ϕ_A) and guarantee (ϕ_G) formulas. We can model the power availability (power fully available - *pfA*) and frequency violation (*vio*) as chance constraints:

$$\begin{aligned} \text{vio}_t &= |50 - f_t^{SYS}| > 0.2 \\ \text{pfA}_t &= |\mathcal{P}_t^{SYS} - P_{\text{target}}| < 0.1 \end{aligned}$$

Here, f_t^{SYS} is the current system frequency and \mathcal{P}_t^{SYS} is the total power output. We make no assumptions for the system to function properly. Thus the system assumption is always fulfilled:

$$\phi_A = \top$$

For simplicity, each property is given indexed by a moment in time in which it should hold. The referenced time is always the interval given by the last operator. Ideally, the desired system property could be expressed in a single guarantee:

$$G_{[0, \infty]} \neg \text{vio}_t U G_{[t+30, t+930]} \text{pfA}_t^{0.95}$$

This formula holds in two cases. When no frequency violation ever occurs $\neg \text{vio}_t$ always holds. Once a violation occurs $\neg \text{vio}_t$ no longer holds and $\text{pfA}_t^{0.95}$ must hold

in the interval $[t + 39, t + 930]$. This second part means that 30s until 930s after the violation the target power is put out with probability of at least 95%. The issue with this formulation is, that existing solvers only support bounded time properties. This can be alleviated by a workaround. If we separate out the violation detection system and only start the frequency response mechanism once a violation is detected then that system can always start at the fixed time $t = 0$ making it a bounded time property. Thus our system guarantee becomes:

$$\phi_G = \neg \text{vio}_0 UG_{[30,930]} pfa_t^{0.95}$$

We may then evaluate if this property holds for some system M , with regards to its variables V , written as $M \models (V, \phi_A, \phi_G)$.

To apply StSTL to open systems we face another challenge. There are no operators to quantify over components. To reason about system guarantees in a changing component space we have to extend the logic.

7.5.3 Component Aggregation Operators

To make contracts scale regardless of the number of components, we establish generalised relationships between their variables. Operators between a system and its subcomponents are called **vertical operators**. Operators between components of the same level are **horizontal operators**. All vertical and horizontal operators are of the form $\text{OP}(V^{SYS}; V^{comp})$ and $\text{OP}(V^{comp_1}; V^{comp_2})$ respectively where V^{SYS} and V^{comp} are lists of system and component level variables. The precise definition of the relation is then given by the semantics of the operator.

In the following, we show the kind of operators we want to introduce to StSTL based on the example system. Satisfaction of a contract is expanded to include the set of operators O on the system: $M \models (V, \phi_A, \phi_G, O)$

For simplicity, we start with a single type of component. These are batteries, characterised by a maximum capacity $E_{max}^{bat,i}$ and maximum power output $P_{max}^{bat,i}$ where i is the index of the given battery. Additionally, we assume that providing FRR is only a secondary use-case. Over the day, a battery is charged and discharged based on the various system loads and energy generation. Stored energy in excess of their main purpose is called **flexibility** $flex^{bat,i}$ and may be used to provide FRR. The load profile (i.e. the load on the battery over time) may be modelled as a stochastic process subsuming the uncertainties of generation and consumption (becoming the environment process in the final model).

Before trying to answer if these batteries can fulfil the system contract, we start with a simpler necessary precondition: Is there enough excess flexibility at a given point in time? We can write this as a chance constraint:

$$flex_t^{[p]} = \mathcal{P}\{flex_t^{SYS} \geq P_{target} \cdot 900\} \geq p$$

To evaluate this chance constraint we need to establish the relation between the system level flexibility $flex^{SYS}$ and the component level flexibilities $flex^{bat,i}$. Clearly, $flex^{SYS}$ is the sum of all $flex^{bat,i}$. We introduce the sum relationship as an operator:

$$\text{SUM}([flex^{SYS}]; [flex^{bat}]) \Leftrightarrow flex_t^{SYS} = \sum_{i=0}^n flex_t^{bat,i}$$

The grid frequency is the same no matter what point it is measured on, thus we can define:

$$\text{EQ}([f^{SYS}]; [f^{bat}]) \Leftrightarrow \forall i \in [1, n] f_t^{SYS} = f_t^{bat,i}$$

Even when enough flexibility is present in the system, we still need to coordinate which battery should provide how much power at a given time. In a swarm-based solution, agents would communicate the desired system property (P_{target}) or some indicator for it with each other directly or through their environment (e.g. with some extra system variable). We assume that the aggregate power output is known and shared between batteries:

$$\begin{aligned} \text{SUM}([P^{SYS}]; [P^{bat}]) &\Leftrightarrow P_t^{SYS} = \sum_{i=0}^n P_t^{bat,i} \\ \text{EQ}([P^{SYS}]; [P^{agg}]) &\Leftrightarrow \forall i \in [1, n] P_t^{SYS} = P_{agg,t}^{bat,i} \end{aligned}$$

We could have made information about the aggregate power decentralised by only defining it as the battery level variable P_{agg}^{bat} and holding that equal between components.

With this we can define a naive control strategy as the battery contract. Whenever there is a frequency violation and the output power is lower/higher than the target power and the battery has flexibility to spare it will output its maximum power for a time step or reduce its output to zero, respectively. We can define the necessary properties as:

$$\begin{aligned} high_t &= P_{agg,t} \geq P_{target} + 0.1 \\ low_t &= P_{agg,t} \leq P_{target} - 0.1 \\ have_flex_t &= flex_t^{bat} \geq P_{max}^{bat} \\ providing_t &= P_t^{bat} > 0 \\ output_zero_t &= P_t^{bat} \leq 0 \\ output_max_t &= P_t^{bat} \geq P_{max}^{bat} \end{aligned}$$

We encode the decrease (dec_P) and increase (inc_P) of the power output based on the system frequency and use them to define the battery guarantee. Once again, we make no assumptions ($\phi_{A,bat}$) in the contract and ensure that the guarantee ($\phi_{G,bat}$) is a bounded time property by moving the violation detection to a separate system and always starting the power output at time 0.

$$\phi_{A,bat} = \top$$

$$\text{dec_P} = \text{high}_t \wedge \text{providing}_t \rightarrow \text{output_zero}_{t+1}$$

$$\text{inc_P} = \text{low}_t \wedge \text{have_flex}_t^{[0,95]} \rightarrow \text{output_max}_{t+1}$$

$$\phi_{G,\text{bat}} = \neg \text{vio}_0 UG_{[30,930]} \text{dec_P} \wedge \text{inc_P}$$

This is obviously a bad control strategy but depending on the number and sizes of batteries it can provide a solution to the system contract and thus is fit for investigation.

7.6 Next Steps

After defining operators to collect aggregated information on contracts and establishing an example system, the next step will be to integrate these operators into the verification process. This includes answering how we can do contract refinement and composition and finally verify systems like these. Similar to the original contract verification process we need to compute the composite contract of the components and show that it refines the system contract. The challenge is deriving the composite of an unspecified and changing number of components.

In that process the example system should be changed to more closely resemble the requirements of a real FRR provider. This includes scaling the power output to the size of the frequency violation, allowing negative power output (i.e. loading a battery) and defining a workable control strategy. Further operators will be included as necessary.

Once this is complete further examples should be found and the verification results must be tested against simulations and real world tests to critically examine if the assumptions made hold in a productive environment.

7.7 Conclusion

In this paper we presented an extension to StSTL with operators that allow the aggregation of information over a variable number of components. This is a step towards the verification and synthesis of heterogeneous stochastic open systems.

The motivating use-case behind this are the demands of the changing energy grid landscape to be more decentralised and allow smaller participants. This in turn invites the use of decentral control structures, promising better scalability and making the system self-adaptive and self-organising in the process.

Under the umbrella of Organic Computing, many concrete control solutions for this type of system are proposed and investigated. The increase in complexity on critical infrastructure must be mitigated by ensuring correct system behaviour. In a lot of cases this is done through simulation-based validation and testing. These give valuable insights but are prone to missing edge cases that may be caught by verification.

While the main motivation for this work is from the energy domain, the developed modelling and verification framework should not be domain specific. It could also

be used for heterogeneous robot swarms, scalable contracts for embedded systems and others. Our aspiration is that a sufficiently general verification framework will be applied to many OC systems, establishing them as both viable and trustworthy solutions for complex problems.

References

1. Cavalcanti, A., Miyazawa, A., Sampaio, A., Li, W., Ribeiro, P., Timmis, J.: Modelling and verification for swarm robotics. In: International Conference on Integrated Formal Methods. pp. 1–19. Springer (2018)
2. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: International School on Formal Methods for the Design of Computer, Communication and Software Systems. pp. 53–113. Springer (2011)
3. Guelev, D.P.: Probabilistic neighbourhood logic. In: International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems. pp. 264–275. Springer (2000)
4. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. *Formal aspects of computing* 6(5), 512–535 (1994)
5. Konur, S.: A survey on temporal logics. arXiv preprint arXiv:1005.3199 (2010)
6. Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems* 60(2), 199–213 (2012)
7. Kouvaros, P., Lomuscio, A.: A cutoff technique for the verification of parameterised interpreted systems with parameterised environments. In: Twenty-Third International Joint Conference on Artificial Intelligence (2013)
8. Kouvaros, P., Lomuscio, A.: A counter abstraction technique for the verification of robot swarms. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
9. Kouvaros, P., Lomuscio, A.: Verifying fault-tolerance in parameterised multi-agent systems. In: IJCAI. pp. 288–294 (2017)
10. Lehnhoff, S., Klingenberg, T., Blank, M., Calabria, M., Schumacher, W.: Distributed coalitions for reliable and stable provision of frequency response reserve. In: 2013 IEEE International Workshop on Intelligent Energy Systems (IWIES). pp. 11–18. IEEE (2013)
11. Leofante, F., Schupp, S., Abraham, E., Tacchella, A.: Engineering controllers for swarm robotics via reachability analysis in hybrid systems. In: ECMS. pp. 407–413 (2019)
12. Lomuscio, A., Pirovano, E.: Verifying emergence of bounded time properties in probabilistic swarm systems. In: IJCAI. pp. 403–409 (2018)
13. Lomuscio, A., Pirovano, E.: A counter abstraction technique for the verification of probabilistic swarm systems. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. pp. 161–169. International Foundation for Autonomous Agents and Multiagent Systems (2019)
14. Müller-Schloer, C., Tomforde, S.: *Organic Computing - Technical Systems for Survival in the Real World*. Birkhäuser (2017), <https://doi.org/10.1007/978-3-319-68477-2>
15. Nafz, F., Steghöfer, J.P., Seebach, H., Reif, W.: Formal modeling and verification of self-* systems based on observer/controller-architectures. In: Assurances for Self-Adaptive Systems. pp. 80–111. Springer (2013)
16. Nuzzo, P., Li, J., Sangiovanni-Vincentelli, A.L., Xi, Y., Li, D.: Stochastic assume-guarantee contracts for cyber-physical system design. *ACM Transactions on Embedded Computing Systems (TECS)* 18(1), 1–26 (2019)
17. Pandya, P.K.: Interval duration logic: Expressiveness and decidability. *Electronic Notes in Theoretical Computer Science* 65(6), 254–272 (2002)

18. Sadigh, D., Kapoor, A.: Safe control under uncertainty with probabilistic signal temporal logic. In: Proc. of Robotics: Science and Systems (2016)
19. Sangiovanni-Vincentelli, A., Damm, W., Passerone, R.: Taming dr. frankenstein: Contract-based design for cyber-physical systems. *European journal of control* 18(3), 217–238 (2012)
20. Schellhorn, G., Tofan, B., Ernst, G., Pfähler, J., Reif, W.: Rgitl: A temporal logic framework for compositional reasoning about interleaved programs. *Annals of Mathematics and Artificial Intelligence* 71(1-3), 131–174 (2014)
21. Tomforde, S., Hähner, J., Müller-Schloer, C.: Incremental design of organic computing systems-moving system design from design-time to runtime. In: International Conference on Informatics in Control, Automation and Robotics. vol. 2, pp. 185–192. SCITEPRESS (2013)
22. Tomforde, S., Müller-Schloer, C.: Incremental design of adaptive systems. *J. Ambient Intell. Smart Environ.* 6(2), 179–198 (2014), <https://doi.org/10.3233/AIS-140252>
23. Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., Schmeck, H.: Observation and control of organic systems. In: *Organic Computing - A Paradigm Shift for Complex Systems*, pp. 325–338. Springer (2011), https://doi.org/10.1007/978-3-0348-0130-0_21
24. Van Hung, D., Zhang, M.: On verification of probabilistic timed automata against probabilistic duration properties. In: 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007). pp. 165–172. IEEE (2007)

A Framework for a Component-based Comparison of Metaheuristics

Helena Stegherr

OrcidID 0000-0001-7871-7309

Organic Computing Group, Augsburg University, Augsburg, Germany

helena.stegherr@informatik.uni-augsburg.de

Abstract. Metaheuristics present useful tools to approximately solve complex optimisation problems. They are applied when mathematical optimisers either cannot be used because no exact definition of the problem is known or their application is too time-consuming. To develop new metaheuristics, nature is often used as a source of inspiration. However, the number of these nature-inspired approaches is growing and it is increasingly difficult to keep an overview of novel ideas and to find the most suitable metaheuristic for the problem at hand. Additionally, with increasing problem complexity, it is necessary to make metaheuristics more efficient, for example by the parallelisation of expensive computational steps or by the adaptation of exploration and exploitation behaviour of the metaheuristic. Analysing metaheuristics with a focus on solving these problems requires an adequate framework for testing and evaluation. In this paper, the necessities of such a framework are described: To analyse functional components of metaheuristics with respect to their effects on optimisation, each metaheuristic has to be modelled in terms of these components. This will also facilitate the detection of novelty in metaheuristics, as components can be abstracted and compared, and even the construction of new metaheuristics from these components. The evaluation of different parallelisation strategies is another important feature the framework should provide. However, most interesting, especially for the field of Organic Computing, is that a component-based design of metaheuristics allows for adaptive changes of these components during runtime, depending on exploration and exploitation requirements.

Keywords: Metaheuristics, Stochastic Optimisation, Nature-inspired Algorithms, Optimisation Frameworks.

8.1 Introduction

Complex optimisation tasks present reoccurring problems in real life as well as in many research areas, including the field of *Organic Computing* (OC) [47]. Many of these problems cannot be solved by exact mathematical optimisers (that terminate after finding the optimal solution) in acceptable time. Thus, stochastic optimisation techniques like heuristics and metaheuristics are needed to approximate solutions for these complex optimisation tasks.

However, the *No-Free-Lunch* theorem states that no metaheuristic performs equally well on all problems [50]. In order to develop enough different but efficient metaheuristics, nature is often used as a source of inspiration. By 2017, there were already almost 200 “different”¹ nature-inspired metaheuristics [37] and by 2020 over 300 [30]. This makes it exceedingly difficult to keep an overview of their functionality, their capabilities, and the problems they have been applied to. Furthermore, unique characteristics and novelty in the approaches are hard to detect, especially under consideration of the definition of the term metaheuristic by Sörensen and Glover in 2013 [42]:

A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. The term is also used to refer to a problem-specific implementation of a heuristic optimization algorithm according to the guidelines expressed in such a framework.

It is necessary to distinguish between novel metaheuristic algorithms and frameworks, as new algorithms are much easier to design but are not as comprehensive as frameworks when considering new insights and applicability [27]. Software frameworks for the evaluation of metaheuristics often incorporate the metaheuristic frameworks and their different algorithmic variations. In the following, we will use the term metaheuristic to imply metaheuristic frameworks, while framework is used in a software-related sense.

Finding the most suitable metaheuristic for the problem at hand is a difficult task, especially for black box problems with unknown and complex fitness landscapes, which are common to OC systems. Furthermore, these problems possess many different characteristics. They can be real-valued or integer problems, static or dynamic, single- or multi-objective, and even distributed [31]. This makes the selection of an appropriate metaheuristic an important task. OC systems often utilise metaheuristics as part of the *Multi-Level Observer/Controller framework*, for example in the *Organic Traffic Control system* [36] or the *Organic Network Control system* [46]. In terms of self-adaptivity in OC systems, metaheuristics are utilised to provide optimal system parameters at runtime [16] and to establish self-adaptation when the fitness landscape of the system is highly dynamic [11]. Again, the most suitable metaheuristic for the specific situation has to be found, together with an appropriate configuration for it.

To alleviate these difficulties, it is beneficial to ascertain the individual components’ performance impact on specific optimisation problems. The basis for this is a framework in which metaheuristics can be assembled by basic functional structures. This framework has to be both conceptual and computational. The conceptual part has to allow for analysing the functional components of metaheuristics that are relevant for the search process. The computational part has to enable the construction of metaheuristics from these components and evaluate their capabilities in terms of search behaviour and performance. Thus, to distinguish novel approaches from established metaheuristics, the components used in their construction can be compared. In

¹ Sörensen questions the amount of novelty in many new metaheuristics and expects some repetition of ideas disguised by the prevalent use of metaphor-specific vocabulary [44].

addition, the impact of certain components can be investigated by their integration or substitution in the metaheuristic when solving the same problem. However, this approach needs a detailed analysis of metaheuristics to identify these basic components. After the analysis of the components, it is possible to utilise the information as a basis for the development of adaptation strategies to select the best components depending on the optimisation problem, or even during the optimisation process. Another important feature of the framework is the possibility of parallelisation. With growing problem complexity, the runtime of metaheuristics is increasing. Thus, parallelisation and distribution of the computation is a necessity. Additionally, it has to be possible to provide different strategies for this task, as some parallelisation strategies can be more efficient for certain metaheuristics or problems. A comparison of the capabilities of these strategies for the included metaheuristics has to be available. Furthermore, the framework will have to incorporate extensive capabilities for experiment design and evaluation.

This paper presents the necessary capabilities that should be included in a framework that allows for a detailed analysis of metaheuristics. It entails the adequate comparison of metaheuristics according to standardised testing and evaluation procedures, the possibility of a component-based analysis and the assessment of the efficiency of different parallelisation strategies. Section 8.2 provides an overview of metaheuristic research on component-based analysis, parallelisation and testing and evaluation of metaheuristics, including aspects of research that still need to be addressed. Sections 8.2.2 and 8.2.3 elaborate on existing metaheuristic frameworks and the necessity to develop yet another one. The concept for a component-based framework is presented in Section 8.3. It provides information on the conceptual considerations when analysing metaheuristics in terms of their components (Section 8.3.1). Afterwards, the basic features the framework needs are presented, as well as the specific additions which have to be made to provide the capabilities for informative comparisons, component analysis and parallelisation (Section 8.3.2). Section 8.4 provides information of further requirements of the framework and Section 8.5 summarises the findings.

8.2 Related Work

This section presents an overview on metaheuristic research and open areas related to the comparison and evaluation of metaheuristics. Existing frameworks for testing metaheuristics will be presented and their advantages and disadvantages will be summarised. This will be utilised in deciding if a new framework is necessary or if an existing framework is worth extending.

8.2.1 Metaheuristic Research

The first step towards constructing a component-based metaheuristic framework is determining what can be considered as a component. A generic framework that categorises components according to their general effects in the metaheuristic is

presented by Bandaru and Deb [5]. They structure the main components of a metaheuristic algorithm into five groups: the *selection plan*, *generation plan*, *replacement plan*, *update plan*, and *termination plan*. Each of those can contain one or more specific operators. Krawiec et al. provide a set of metaheuristic design patterns which provide an abstraction of metaheuristics and help to identify the functional components [24]. More direct descriptions of possible components are also available. An analysis of components relating to the intensification and diversification capabilities of metaheuristics is presented by Blum and Roli [9]. In addition, Lones provides a component-based classification, focusing on the search heuristics the metaheuristic incorporates [26, 27].

The testing and evaluation of the performance of metaheuristics is extremely important. Thus, the experimental design and the applied statistical methods have to follow standardised procedures. A standard for the design of experiments is presented by Hooker [18, 19] and Barr et al. [6]. Further information with a more specialised view on metaheuristics is provided by Birattari et al. [8] and Rardin and Uzsoy [38]. García et al. present statistical methods for the evaluation of metaheuristic performance [17]. These are complemented by Eftimov et al., who developed a more significant statistical evaluation [15].

Detailed information on parallel metaheuristics is provided by Alba et al. [2, 4]. They present not only different strategies and examples, but also detailed information on the evaluation of these parallel approaches [3].

In terms of more specific research on metaheuristics and performance comparisons, some categories of metaheuristics have been analysed more extensively, resulting in an increased publication of methods [30]. Bio-inspired metaheuristics, especially *evolutionary algorithms* and *swarm-based metaheuristics*, are in the focus of comparisons and evaluations [1, 7, 41]. Next to those comparisons, which are mostly done using benchmark problems, metaheuristics have also been utilised in complex real-world applications, for example for data mining, scheduling, and image processing [21]. Furthermore, there are some general analyses of intensification and diversification (or exploitation and exploration) in metaheuristics [51, 52], with additional, more detailed evaluations in established metaheuristics, for example in *swarm-based methods* [22, 39] and *evolutionary algorithms* [12].

Though metaheuristics have been researched for years, there are still open questions and problems to be solved. For some of those, a component-based framework as proposed in this paper and the theoretical considerations behind it provide an appropriate approach. These include [13, 33, 43]:

- adequate and standardised testing protocols, including methodologies for comparisons and the selection of appropriate benchmark problems
- a coherent implementation without focusing on metaphor-related vocabulary and with an accessible source code
- a focus on better understanding and improvement of existing methods, especially in terms of intensification and diversification (or exploitation and exploration) behaviour and parallelisation strategies

8.2.2 Metaheuristic Frameworks

This section first provides information on commonly used frameworks for evaluating metaheuristics. Their major features and the resulting advantages and disadvantages are summarised in Table 8.1.

A comprehensive summary and a detailed comparison of some of these frameworks was presented by Parejo et al. in 2011 [35]. Furthermore, they developed their own object-oriented framework, FOM, for the development and implementation of metaheuristics [34]. An important aspect of FOM is the separation of the problem from the algorithms. This leads to a reusability of metaheuristic components in different problems.

Another early but extensive framework is ParadisEO [10]. It extends the EO library for evolutionary computation [23]. ParadisEO is focused on a parallel and distributed design of metaheuristics and their hybridisation. Another important feature is its flexibility concerning data representation and operators. An extension for multi-objective optimisation was developed in 2010 [25].

HeuristicLab presents a versatile framework which is still under active development [48, 49]. The optimisation process is abstracted to provide a universal, extensible and paradigm-independent optimisation environment, without focusing on any specific kind of optimisation algorithm or problem. Additionally, the reuse of algorithms and problems is facilitated and adding new ones is straightforward.

ECJ started as an evolutionary computation library [29, 40]. It provides advantages in genetic programming, massive distributed computation and coevolution. The framework was further extended into a toolkit, including many metaheuristics. However, only evolutionary-based metaheuristics are integrated into ECJ.

The opt4j framework is a modular framework for solving complex optimisation tasks [28]. Problems are decomposed into heterogeneous subtasks that require concurrent optimisation.

jMetal is a framework designed focusing on multi-objective optimisation [14]. It is based on an object-oriented architecture and aims at being easy to use, flexible and extensible. jMetal was re-designed in 2015 and now makes use of design patterns [32]. Furthermore, it provides parallel execution and live interaction with running algorithms.

8.2.3 Is there a Need for a New Framework?

With several established and comprehensive frameworks, the question arises, if another one is necessary. However, none of the existing frameworks directly fits the requirements for a component-based analysis of metaheuristics and additionally provides the means for extensive parallelisation studies. Furthermore, extensibility of the framework has to be possible in terms of adding new metaheuristics, components, parallelisation strategies, and optimisation problems. Especially for the application in real-world OC systems, the framework must be able to include complex real-world problems. Additionally, combining the included metaheuristics with rule-based learning systems common to OC has to be possible. All this depends on the framework's structure, but also on the quality of its documentation.

Table 8.1. Comparison of commonly used metaheuristic frameworks.

Framework	Language	Specialisation	Advantages	Disadvantages
FOM [34]	Java	reusability	architecture	number of included metaheuristics, not actively maintained
ParadisEO [10, 25]	C++	parallelisation	parallelisation options	number of included metaheuristics, years without development
HeuristicLab [48, 49]	C#	reusability, expandability	architecture, GUI, under active development	number of included metaheuristics, only supports Windows
ECJ [29, 40]	Java	distributed computation	library style	number of included metaheuristics
opt4j [28]	Java	problem decomposition	GUI	number of included metaheuristics, not actively maintained
jMetal [14, 32]	Java	multi-objective	specialisation on multi-objective, under active development	number of included metaheuristics

Each framework is specialised on performing one specific task but lacks sufficient support for other approaches. While capabilities for parallelisation are given in many frameworks, for example ParadisEO, ECJ, jMetal and HeuristicLab, a component-based structure in the required design is not available, though HeuristicLab and FOM are based on an at least similar idea. Implementing this component-based design within an existing extensive framework is immensely complicated and time-consuming. However, designing this kind of framework from scratch also is. Another problem is the number of pre-implemented metaheuristics. All frameworks only include established methods, at most about 20 different metaheuristics. Thus, adding more metaheuristics will be necessary, but is again time-consuming and requires detailed knowledge of the framework.

These problems complicate the extension of existing frameworks too much and make a more directly suitable novel approach necessary. However, all frameworks still need to be reviewed in detail as they incorporate many features that should and will be reused. ParadisEO provides the most extensive parallelisation capabilities that will also be included in the novel framework. HeuristicLab, FOM and jMetal have a similar structure as required by the new framework, though they do not implement the components exactly the way required. The GUI of HeuristicLab is another feature that will inspire the new implementation, though it will be extended. To facilitate the combination with rule-based learning, the implementation of the new framework as a library similar to ECJ, but with the option of using it with a GUI, is intended.

8.3 Concept of a Component-based Framework

The identification, evaluation and (adaptive) recombination of relevant metaheuristic components is one of the primary goals the framework has to accomplish. Another goal is the application and analysis of different parallelisation strategies. Thus, the user has to be able to interact with the framework in the following ways:

- The user will be able to specify the metaheuristic that should be used, or the components it should contain, and apply it on the problem at hand.
- To determine which components could be relevant to this problem, the user can draw on the information from the previous component analyses.
- For evaluating or comparing different algorithms on benchmark optimisation problems, the user can utilise the framework's GUI.
- Furthermore, a specification of the desired parallelisation strategy is possible.
- When the metaheuristics should be applied in a larger existing system, its library-like style will allow for their utilisation.

To this end, we first have to differentiate the conceptual considerations necessary for the framework development from the computational aspects. The conceptual framework is concerned with the analysis of metaheuristics in terms of their components. The computational framework will enable comparisons and performance evaluations of metaheuristics by utilising the established component structure.

8.3.1 Conceptual Framework

Before implementing metaheuristics in a component-based way, it is imperative to define those components. Blum and Roli describe their I&D components as “any algorithmic or functional component that has an intensification and/or diversification effect on the search process” [9]. In addition, Lones identifies common metaheuristic concepts and general approaches that provide mutual high level categories for the components. Thus, we will use the term component for all general concepts inherent to metaheuristics that can influence the search behaviour in any way.

Finding and describing all necessary components still requires a detailed analysis of all metaheuristics that should be included in the framework. For common metaheuristics, there already are detailed descriptions including at least their basic components [45]. Less known metaheuristics or new approaches, however, need to be reviewed individually. Components found are matched against the intensification and diversification components as described by Blum and Roli [9] and the heuristic search components described by Lones [26, 27]. Metaheuristic design patterns can also help to identify and describe these components [24]. Additionally, functional structures not previously described as metaheuristic components have to be examined.

An exemplary (but presumably still incomplete) component analysis for the *Genetic Algorithm* is shown in Figure 8.1. The main components identified so far are *Selection*, *Crossover*, *Mutation*, *Population Generation* and *Replacement*. These fit the definitions of Blum and Roli [9] and Lones [26, 27] as they all exhibit influences on intensification and/or diversification behaviour and can be categorised

into common search structures. Additionally, each main component can occur in implementation-specific manifestations. For example, a mutation is performed as a Gaussian perturbation, an insertion, an inversion, or one of many other possible procedures. These specific options have to be chosen depending on the problem. However, a self-adaptive component selection can lead to more efficient search behaviour.

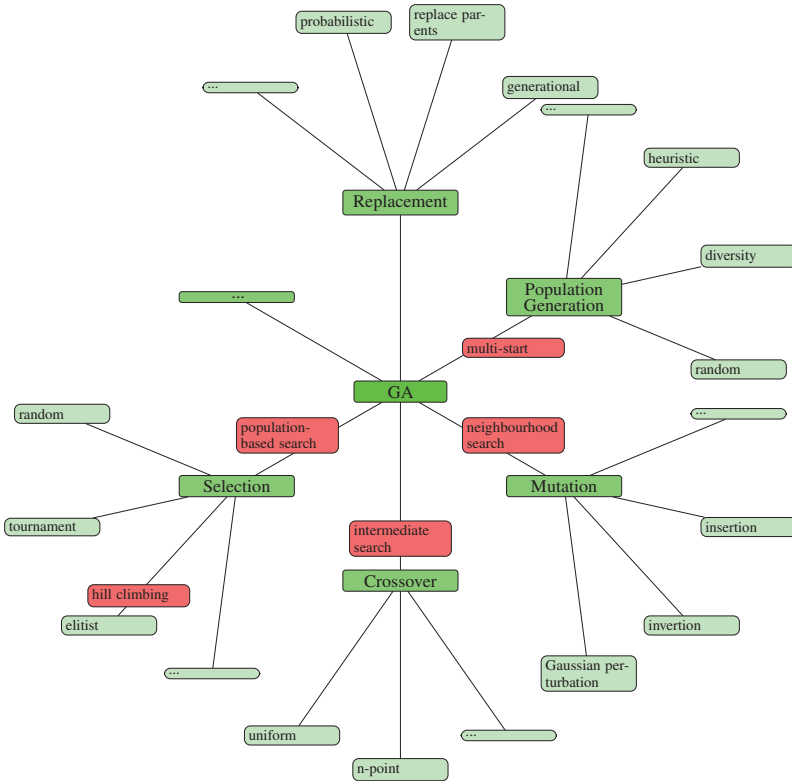


Fig. 8.1. Example of components of the *Genetic Algorithm* and the higher level categories (depicted in red) developed by Lones [26,27]

8.3.2 Computational Framework

The computational part of the framework is based on the determined component structure, therefore enabling a component-based implementation of metaheuristics. Its general structure will be similar to the generic framework described by Bandaru and Deb [5]. This allows for a consistent implementation of all metaheuristics. It also facilitates the addition or exchange of components within one metaheuristic, but also between different metaheuristics. The computational framework must furthermore

provide the means for standardised testing and evaluation of metaheuristics, their comparison and an analysis of their components' behaviour. Parallelisation of all included metaheuristics must also be possible, utilising different established strategies. The following sections will provide more details on these specific requirements.

8.3.2.1 Prerequisites

There are some fundamental considerations required for the framework. These include factors that have to be taken into account when implementing, testing and evaluating metaheuristics.

Considerations regarding the implementation, testing and evaluation of metaheuristics all aim at providing meaningful results for the performed analyses. When a (new) metaheuristic is claimed to perform better than another one, there are too often major differences in the amount of adaptation and optimisation put into the metaheuristics [44]. For example, comparing a metaheuristic with tuned parameters against one with basic parameters derived from literature may influence the outcome of the experiment, if the basic parameters are not optimal for the optimisation problem. This kind of bias needs to be avoided. With the knowledge of components inherent to metaheuristics and additional information, e.g., if a metaheuristic is population-based, swarm-based, evolutionary, etc., it is possible to classify metaheuristics according to relevant criteria. It facilitates choosing suitable metaheuristics, for example if it is of interest to find the best population-based algorithm with certain components for a specific problem class. Assume, for example, that two population-based metaheuristics are compared in terms of the best solution found on a specific problem. Both metaheuristics utilise crossover as one of their components. However, the crossover operator of one metaheuristic is uniform crossover, while in the other it is n-point crossover. To ascertain equal chances for a good performance, one has to at least test what happens if both utilise the same operator.

A difficulty for both, the component analysis and the classification, lies in the algorithmic descriptions of most nature-inspired metaheuristics [44]. There is no standardised notation and inspiration-related descriptions are commonly used. Therefore, it is necessary to analyse these descriptions to detect if they relate to the same concepts and components.

Of less importance, but also to be taken into consideration, are implementation basics, for example the programming language and the documentation. The framework will presumably be implemented in either Rust or Scala. Both programming languages facilitate building and managing larger projects and both are suitable for parallelisation approaches. The framework and its functionalities have to be well documented, as it will be available open source and therefore should be adaptable and extensible by all users.

8.3.2.2 Implementation of Metaheuristics

With the focus on finding metaheuristic components with a high effectiveness on specific problems, the framework should include as many metaheuristics and, thus, as

many components as possible. Established metaheuristics, e.g. the Genetic Algorithm, Particle Swarm Optimisation, Simulated Annealing and Ant Colony Optimisation, have to be integrated in a way that allows for fast and easy application. All metaheuristics will be based on reusable components. These components are presented in a way that allows for exchanging components of the same type. This applies especially for search components as described by Blum and Roli which are included in many different metaheuristics [9].

Furthermore, some more basic aspects have to be considered. All metaheuristics have to be adaptable to different problems, e.g., combinatorial problems and problems with real-valued solutions. This also includes that if a new optimisation problem is added, the metaheuristics have to be applicable to this problem as well. Additionally, it has to be possible to add new metaheuristics, as combination of already included components as well as new components. This process has to be as simple as possible.

8.3.2.3 Evaluation

Testing and evaluation have to follow standardised procedures. Thus, these procedures will be included in the framework, from the design of experiments to the statistical evaluation. Experimental setups have to be prepared to facilitate the tests. Suitable statistical methods also have to be included. However, it has to be possible to add new experimental designs and the corresponding statistics, either if standards change or if the researcher has a specific but not previously anticipated goal in mind. Finally, adequate measures for the presentation of the results have to be prepared. This includes the format of the output, for example as detailed tables, all additional information on the statistical evaluation and corresponding graphical plots of the data.

8.3.2.4 Comparison

The framework has to allow for all metaheuristics to be compared on a range of problems. Several metaheuristics can be selected and the evaluation is based on the appropriate statistical methods. Additionally, if metaheuristics have extensions or adaptations, it has to be possible to select the respective version. The parameters necessary for the experiments, the metaheuristic parameters as well as which statistical methods to apply and how to format the output, are also of importance. The framework has to present a set of standard parameters, but they have to be adaptable for the respective application.

8.3.2.5 Component Analysis

The aim is the analysis of exceptional components and their effects on optimisation. Especially the effects on intensification and diversification behaviour and the overall distribution of solutions in the search space will be analysed. Furthermore, it is of interest how they can be recombined to present new strategies applicable to specific problems or problem classes. The framework has to provide the option of choosing

metaheuristics for comparisons according to the components they contain. Additionally, it has to be possible to exchange components in a metaheuristic and compare it against the basic version. This might not be possible for every component in every metaheuristic, but will lead to detailed knowledge of the components' performance on the respective problem. Another factor that has to be considered are dependencies between components. These have to be analysed conceptually, but it must also be possible to evaluate them in the computational framework.

8.3.2.6 Parallelisation

For metaheuristic parallelisation studies, two main aspects have to be considered. One is the algorithmic step that should be parallelised. Commonly, this step is the calculation of the function value of the current solution, as this is usually the most time-consuming part. However, parallelisation is also possible at other stages. Furthermore, the method for parallelisation has to be defined. It can, for example, be actor-based or based on multi-threading. In addition, especially for population-based metaheuristics, there are several designs for parallelisation, such as the master-slave model or the multiple-population model [2]. The framework has to be able to parallelise the metaheuristics in different ways and compare the results for specific problems.

8.4 Further Extensions

Further extensions are necessary to make the computational framework generally applicable to the analysis of metaheuristics and for finding the best metaheuristic for a given problem. One of those is the optimisation of hyperparameters. As the performance of a metaheuristic strongly depends on its parameter settings, methods for optimising these parameters have to be available in the framework. These will include the most common tuning approaches [20], but the framework has to enable adding new or individual methods. Furthermore, basic parameter settings from literature have to be available if a comparison with these results is required or only preliminary tests are performed.

Another extension for the framework is the inclusion of multi-objective optimisation problems. This is especially important when utilising the framework to solve complex real-world problems which are common to the field of OC. The framework has to provide implementations of the metaheuristics capable of multi-objective optimisation or facilitate the adaptation of the algorithms.

To enhance the applicability of the framework to problems from the field of OC, but also to make use of some of the basic OC principles, the self-adaptive selection of metaheuristic components will be considered as a major feature of the framework. First, however, the intensification and diversification capabilities of the components have to be evaluated. Furthermore, an analysis of their influence on the distribution of the solutions in the search space is required, especially for population-based metaheuristics. The results of both studies are problem-dependent and require

generalisation, for example by analysing the behaviour over many different problems. With this information as a knowledge-base, it is possible to derive adaptation strategies for the usage of components in different situations of the search process. These strategies consist of rules that will be predefined (or even learned) and allow for the self-adaptation of the component composition during the search. Adaptation strategies can also benefit from research on hyperheuristics. Hyperheuristics select the best heuristic for an optimisation problem, while the component adaptation would select the best combination of components within a metaheuristic. However, with a general framework implementing the metaheuristics, it is possible to even adapt components during the optimisation process.

8.5 Conclusion

There are still many open questions in metaheuristic research. However, there are too many new nature-inspired metaheuristics to keep an overview of their potential and novel ideas. The analysis of metaheuristics needs to be extended but also facilitated to cope with these and still not miss any important development. To this end, metaheuristic frameworks provide help. They include different algorithmic approaches and the adequate statistical tools for their evaluation. It is also essential that metaheuristics which are compared are implemented in the same language and evaluated on the same system. General metaheuristic frameworks facilitate this and standardise the procedure.

However, the existing frameworks are not directly applicable to every task. They are hard to extend as they either have been developed over years or have long gaps between maintenance times and additions. Furthermore, they are complicated to understand due to their documentation missing important details. Thus, a new concept for a framework for the analysis of metaheuristics was created. This framework is focused on evaluating the influence of metaheuristic components on the overall search behaviour, especially on intensification and diversification. Furthermore, it aims at facilitating the improvement of metaheuristic performance by enabling their parallelisation and self-adaptive selection of components necessary for the given search situation. To this end, the framework includes conceptual considerations, especially in terms of defining and finding the components of a metaheuristic, and computational capabilities to be able to conduct large-scale studies on metaheuristics.

The next steps to realise the proposed framework will include more detailed analyses of metaheuristics in order to determine their components and the implementation of a basic set of metaheuristics. Furthermore, a number of common benchmark problems will be included and the necessary evaluation procedures will be established. Additional optimisation problems and metaheuristics will then be added regularly. Once a suitable scheme for these approaches has been found, the framework will be extended to enable parallelisation, hyperparameter tuning and multi-objective optimisation, as well as first component adaptation approaches.

References


1. Al-Amry, R.A., Al-Gaphari, G.: Survey on Recent Bio-Inspired Optimization Algorithms. *International Journal of Computer Science and Network (IJCSN)* 7(6) (2018)
2. Alba, E.: *Parallel Metaheuristics*. John Wiley & Sons (2005)
3. Alba, E., Luque, G.: Evaluation of Parallel Metaheuristics (2006)
4. Alba, E., Luque, G., Nesmachnow, S.: Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research* 20(1), 1–48 (2012)
5. Bandaru, S., Deb, K.: Metaheuristic Techniques. In: *Decision Sciences*, pp. 693–750. CRC Press (nov 2016)
6. Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G.C., Stewart, W.R.: Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics* 1(1), 9–32 (1995)
7. Binitha, S., Sathya, S.S.: A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSCE)* 2(2) (2012)
8. Birattari, M., Paquete, L., Stützle, T., Varrentrapp, K.: Classification of Metaheuristics and Design of Experiments for the Analysis of Components. Tech. rep., Tech. Rep. AIDA-01-05 (2001)
9. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization. *ACM Computing Surveys* 35(3), 268–308 (2003)
10. Cahon, S., Talbi, E.G., Melab, N.: ParadisEO: a framework for parallel and distributed biologically inspired heuristics. In: *Proceedings International Parallel and Distributed Processing Symposium*. IEEE Comput. Soc (2003)
11. Cakar, E., Tomforde, S., Müller-Schloer, C.: A role-based imitation algorithm for the optimisation in dynamic fitness landscapes. In: *2011 IEEE Symposium on Swarm Intelligence*. IEEE (2011)
12. Crepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms. *ACM Computing Surveys* 45(3), 1–33 (jun 2013)
13. Del Ser, J., Osaba, E., Molina, D., Yang, X.S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P.N., Coello Coello, C.A., Herrera, F.: Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation* 48, 220–250 (2019)
14. Durillo, J.J., Nebro, A.J., Alba, E.: The jMetal framework for multi-objective optimization: Design and architecture. In: *IEEE Congress on Evolutionary Computation*. IEEE (2010)
15. Eftimov, T., Korošec, P., Seljak, B.K.: A Novel Approach to Statistical Comparison of Meta-heuristic Stochastic Optimization Algorithms using Deep Statistics. *Information Sciences* 417, 186–215 (2017)
16. Fredericks, E.M., Gerostathopoulos, I., Krupitzer, C., Vogel, T.: Planning as Optimization: Dynamically Discovering Optimal Configurations for Runtime Situations. In: *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. IEEE (2019)
17. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences* 180(10), 2044–2064 (2010)
18. Hooker, J.N.: Needed: An Empirical Science of Algorithms. *Operations Research* 42(2), 201–212 (1994)
19. Hooker, J.N.: Testing heuristics: We have it all wrong. *Journal of Heuristics* 1(1), 33–42 (1995)
20. Huang, C., Li, Y., Yao, X.: A Survey of Automatic Parameter Tuning Methods for Metaheuristics. *IEEE Transactions on Evolutionary Computation* 24(2), 201–216 (2020)

21. Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y.: Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review* 52(4), 2191–2233 (2018)
22. Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y.: On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications* 31(11), 7665–7683 (jul 2018)
23. Keijzer, M., Merelo, J.J., Romero, G., Schoenauer, M.: Evolving Objects: A General Purpose Evolutionary Computation Library. *Artificial Evolution* 2310, 829–888 (2002)
24. Krawiec, K., Simons, C., Swan, J., Woodward, J.R.: *Metaheuristic Design Patterns: New Perspectives for Larger-Scale Search Architectures*, pp. 1–36. IGI Global (2018)
25. Liefvooghe, A., Jourdan, L., Legrand, T., Humeau, J., Talbi, E.G.: ParadisEO-MOEO: A software framework for evolutionary multi-objective optimization. In: *Advances in Multi-Objective Nature Inspired Computing*, pp. 87–117. Springer Berlin Heidelberg (2010)
26. Lones, M.A.: Metaheuristics in nature-inspired algorithms. In: *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14*. ACM Press (2014)
27. Lones, M.A.: *Mitigating Metaphors: A Comprehensible Guide to Recent Nature-Inspired Algorithms*. SN Computer Science 1(49) (2019)
28. Lukasiewicz, M., Glaß, M., Reimann, F., Teich, J.: Opt4J. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*. ACM Press (2011)
29. Luke, S.: ECJ then and now. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17*. ACM Press (2017)
30. Molina, D., Poyatos, J., Ser, J.D., García, S., Hussain, A., Herrera, F.: *Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations*. *Cognitive Computation* 12(5), 897–939 (2020)
31. Müller-Schloer, C., Tomforde, S.: *Organic Computing - Technical Systems for Survival in the Real World*. Springer International Publishing (2018)
32. Nebro, A.J., Durillo, J.J., Vergne, M.: Redesigning the jMetal Multi-Objective Optimization Framework. In: *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference - GECCO Companion '15*. ACM Press (2015)
33. Nesmachnow, S.: An overview of metaheuristics: accurate and efficient methods for optimisation. *International Journal of Metaheuristics* 3(4), 320 (2014)
34. Parejo, J.A., Racero, J., Guerrero, F., Kwok, T., Smith, K.A.: FOM: A Framework for Metaheuristic Optimization. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg (2003)
35. Parejo, J.A., Ruiz-Cortés, A., Lozano, S., Fernandez, P.: Metaheuristic optimization frameworks: a survey and benchmarking. *Soft Computing* 16(3), 527–561 (2011)
36. Prothmann, H., Rochner, F., Tomforde, S., Branke, J., Müller-Schloer, C., Schmeck, H.: Organic Control of Traffic Lights. In: *Proceedings of the 5th International Conference on Autonomic and Trusted Computing (ATC-08)*. pp. 219–233. Springer Berlin Heidelberg (2008)
37. Rajpurohit, J., Sharma, T.K., Abraham, A., Vaishali: Glossary of Metaheuristic Algorithms. *International Journal of Computer Information Systems and Industrial Management Applications* 9, 181–205 (2017)
38. Rardin, R.L., Uzsoy, R.: Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial. *Journal of Heuristics* 7(3), 261–304 (2001)

39. Salleh, M.N.M., Hussain, K., Cheng, S., Shi, Y., Muhammad, A., Ullah, G., Naseem, R.: Exploration and Exploitation Measurement in Swarm-Based Metaheuristic Algorithms: An Empirical Analysis. In: *Advances in Intelligent Systems and Computing*, pp. 24–32. Springer International Publishing (2018)
40. Scott, E.O., Luke, S.: ECJ at 20. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '19*. ACM Press (2019)
41. Selvaraj, C., Kumar, S., Karnan, M.: A Survey on Application of Bio-Inspired Algorithms. *International Journal of Computer Science and Information Technologies (IJCSIT)* 5(1), 366–370 (2014)
42. Sörensen, K., Glover, F.W.: Metaheuristics. In: *Encyclopedia of Operations Research and Management Science*, pp. 960–970. Springer US (2013)
43. Sörensen, K., Sevaux, M., Glover, F.: A History of Metaheuristics, pp. 791–808. Springer International Publishing, Cham (2018)
44. Sörensen, K.: Metaheuristics—the metaphor exposed. *International Transactions in Operational Research* 22(1), 3–18 (2015)
45. Talbi, E.G.: *Metaheuristics*. John Wiley & Sons (2009)
46. Tomforde, S., Cakar, E., Hähner, J.: DYNAMIC CONTROL OF NETWORK PROTOCOLS - A New Vision for Future Self-organising Networks. In: *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics* (2009)
47. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic computing in the spotlight. CoRR abs/1701.08125 (2017), <http://arxiv.org/abs/1701.08125>
48. Wagner, S., Affenzeller, M.: HeuristicLab: A Generic and Extensible Optimization Environment. In: *Adaptive and Natural Computing Algorithms*, pp. 538–541. Springer-Verlag (2005)
49. Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., Vonolfen, S., Kofler, M., Winkler, S., Dorfer, V., Affenzeller, M.: *Advanced Methods and Applications in Computational Intelligence, Topics in Intelligent Engineering and Informatics*, vol. 6, chap. Architecture and Design of the HeuristicLab Optimization Environment, pp. 197–261. Springer (2014)
50. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1), 67–82 (1997)
51. Xu, J., Zhang, J.: Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis. In: *Proceedings of the 33rd Chinese Control Conference*. IEEE (jul 2014)
52. Yang, X.S., Deb, S., Fong, S.: Metaheuristic Algorithms: Optimal Balance of Intensification and Diversification. *Applied Mathematics & Information Sciences* 8(3), 977–983 (2014)

Incident-aware Resilient Traffic Management for Urban Road Networks

Ingo Thomsen

✉0000-0002-0850-4786 

Intelligent Systems, Christian-Albrechts-Universität zu Kiel, 24118 Kiel, Germany
int@informatik.uni-kiel.de

Abstract. The control and management of urban road networks can be challenging due to overall rising traffic volumes, fluctuating demands within the network and unforeseen disturbances caused by traffic incidents. This all can lead to congestion problems, which might be difficult to be managed by centralised approaches that only react to the traffic without assessing the underlying incidents. This PhD project engages in the development of the resilient management system *InTURN*. It will define possible traffic incidents and their varying effects. The detection and cooperative validation of these incidents will be used to offer suitable network-wide traffic light signalisation and route recommendations. An explicit goal is the development of a working system within the context of simulated traffic networks and demands.

Keywords: Organic Computing, Traffic Management, OTC, InTURN, Traffic Incident Detection

9.1 Introduction

Present road traffic is increasing due to an unbroken demand for public and individual mobility, which can lead to serious congestion problems in urban traffic. Massive traffic volumes put high demands on management systems due to time-dependent changes. This is intensified by traffic disturbances with varying characteristics (duration, location, severity, ...) and the individual driver behaviour. Established approaches are often limited to reacting only to observed traffic without detecting and assessing underlying incidents.

The project *Incident-aware Resilient Traffic Management for Urban Road Networks* – in short *InTURN* – aims for the development of a self-adaptive and self-organising system (SASO). Such SASO systems tackle the complexity of controlling by distributing the decisions making process among autonomous agents, which in turn cooperate with each other: Goals can be reached in a large system without centralised ruling. As traffic management deals with such a distributed system (see Section 9.2), the incident detection and classification of InTURN is constantly adjusted by an autonomous learning mechanism and supported by cooperative validation of incidents.

This inherently allows for the detection of faulty traffic sensors as well. The starting point for this development is the preliminary work of an integrated management approach:

The Organic Traffic Control (OTC) [7] offers management of traffic light signalisation, route guidance and the formation of progressive signal systems (PSS), which are also called “synchronised traffic lights” or “green waves”. They are the result of coordinating signal phases between neighbouring intersections along roads with high traffic demand. The goal is to reduce the mean waiting times and to minimise stops when traversing the network. A distributed approach, called Decentralised Progressive Signal Systems (DPSS) [11] is used in OTC.

The main contribution of this PhD project towards the envisioned InTURN system is the constant assessment of traffic incidents and the utilisation of resulting insights on multiple levels, ranging from alterations of individual traffic lights to network-wide traffic guidance. For this, novel approaches will be developed, evaluated and combined. That leads to resilient traffic management, and by incorporating the incident assessment the following abilities of the OTC are extended: Self-organised traffic signalisation, provision of progressive signal systems and dynamic route guidance (see also Section 9.2.4).

The remainder of this paper is organised as follows: The next section presents the basic aspects of urban traffic model as used within the bounds of InTURN, together preliminary work of OTC as basis for the development. Section 3 then solely focuses on incidents as an aspect of the model. This is followed by a depiction of the research objectives and resulting challenges in Section 9.4. The actual development tasks and their respective methods are described in Section 9.5. The final Section 9.6 then summarises the scope of envisioned InTURN system.

9.2 Urban Roads Model and Preliminary Work

Traffic control and management in the context of urban road networks is challenging as unforeseeable events and dynamic problems have to be handled. In order to formulate these challenges and associated tasks within the scope of this PhD project, the domain model (urban road networks and traffic demand) is presented below, followed by preliminary work based on it. As incidents are fundamental to the InTURN project, the incident model is described in more detail in Section 9.3.

9.2.1 Road Networks

The first aspect of the domain model are urban road networks itself. In addition to the base idea of intersections connecting sections, each intersection is equipped with industry-standard traffic light controllers (TLC) for each incoming section. They follow a phase-based signalisation schedule as depicted in Figure 9.1.

The incoming and outgoing sections of a intersection are equipped with detectors, that are able to count vehicles. A real-life equivalent could be induction loops. Junctions differ from intersection: Here, a road does not intersect or “cross” another

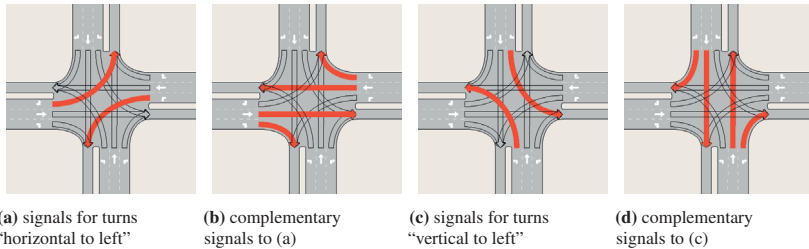


Fig. 9.1. For a signalisation schedule the traffic signals are combined. In case of this exemplary four-armed intersection it could result in the four signal groups (a) to (b), each with non-blocking signals. The groups are combined into a signal plan where the individual phase times add up to the total cycle time. Generally, neighbouring controllers with the same cycle time can be synchronised.

one but merely merges with it. Junctions are not equipped with traffic lights or detectors and are therefore not under control. The traffic lights of an intersection are managed by an OTC controller (see Section 9.2.4) and its detectors form the *sensor horizon*. The controller is aware of its immediate neighbourhood. In addition to the traffic lights, all incoming roads are provided with variable message signs (VMS). These represent real-life LED displays, which are mounted across roads and can be programmed arbitrarily to relay route recommendations or other information to the drivers. The connecting road sections are comprised of one or more lanes with fixed driving directions and speed limits. Only cars and bigger vehicles are considered, and only suitable roads are modelled. Bike lanes or public footpaths are not taken into account. Figure 9.2 shows examples of two types of road networks under consideration: Manhattan-style and more complex networks.

9.2.2 Traffic Demands

The traffic demand forms the second aspect of the model. Using an Origin-Destination matrix is a common approach. Such an O/D-Matrix defines how many *vehicles/hour* traverse from every origin to every destination, without specifying the actual routes through the road network. Instead, the path for each vehicle is chosen by a simulator while heeding a simple directive, e.g., minimising the travelling time. For example, in network of Figure 9.2 the centroids *A* to *L* would each act as origin and destination. The matrix may change during the simulation to represent changes in demand, for instance, due to rush hour. Several traffic demands with and without such changes will be modelled, to act as basis or "ground truth" for investing various incident types described in Section 9.3.

9.2.3 Simulation

Within the scope of InTURN, the road networks, traffic demands and incidents are modelled using a commercial traffic simulation software: *Aimsun Next 20* [1]. The road

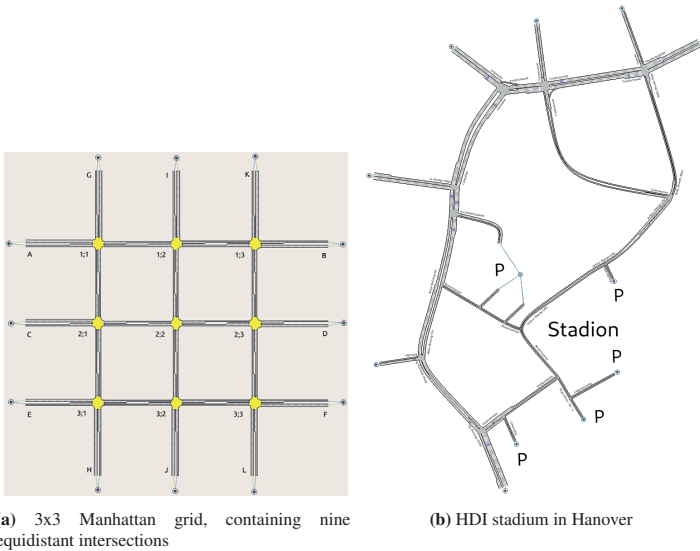


Fig. 9.2. Different types of networks under consideration. Regular, fully connected Manhattan networks are used for the initial development, followed by more complex, real-world models. Generally, sections at the edge are connected to *centroids*, which describe the incoming and outgoing traffic.

networks are designed using the GUI. Then traffic demand and incidents, combined as *experiments*, are simulated by the software. During these simulations, the extensive software API allows for the observation of runtime variables (e.g., detector readings) and control of components (like alterations of TLCs or lane closures). Before resp. after each simulation this is done by a separate management software that interprets this runtime information to determine appropriate alterations of, for example, signal phases or variable message signs.

9.2.4 Organic Traffic Control

A system developed according to the principles of Organic Computing [4] consists of a productive part and a control mechanism based on an observer/controller architecture. It adapts dynamically to the environment it interacts with and exhibits desirable so-called self*-properties: It can be self-organising, self-configuring, self-optimising, self-healing and self-protecting.

The simulated TLCs in this context have fixed signal plans, but although they therefore work autonomously, they do not dynamically adapt to changing traffic conditions. To make the traffic management system aware of impaired road conditions due to incidents, the InTURN system automatically identifies and classifies these incidents. It expands the Organic Traffic Control (OTC), which based on the observer/controller

paradigm above. Each intersection controller is equipped with one adaptation module. The OTC system distinguishes between three interconnected goals:

- Alter TLC settings to accommodate for the current traffic situation and improve this adaptation strategy over time based on reinforcement learning [9].
- Establish progressive signal systems in a self-organised manner using the DPSS [10, 11]: In a synchronised, three-phase process the partners for a green wave are determined. This requires explicit negotiation of partnerships and may utilise additional centralised knowledge.
- Provide drivers through variable message signs with route recommendations that take the current state of the traffic network into account [8].

Figure 9.3 outlines the multilevel design of the OTC system: The simulated traffic light controller constitutes the System under Observation and Control (SuOC) according to Organic Computing. The layer 1 on top of this contains a observer/controller component that monitors the current traffic condition and reconfigures the TLC settings. The appropriate parameter set is chosen using a modified Learning Classifier System (LCS), which is a rule-based reinforcement learning system. Layer 2 then provides an optimisation of the TLC parameters. This is done offline, using an Evolutionary Algorithm (EA) and running a second traffic simulator instance.

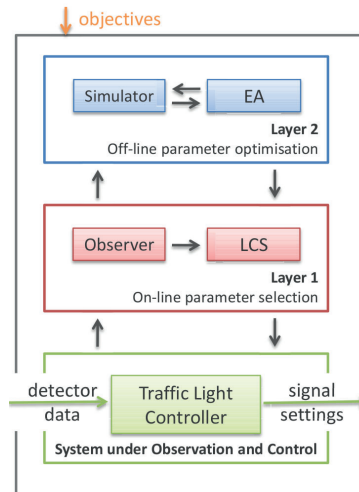


Fig. 9.3. Overview of the multilevel OTC architecture, based on [11]. It consists of the SuOC as well as the online control component at layer 1 and the offline optimisation at layer 2.

One task of the development towards InTURN is an initial migration based on a modified software architecture. On the one hand this compensates for changes of the simulation API. On the other hand it facilitates extended preprocessing necessary for the incident detection and classification.

9.3 Incidents

The model for urban road traffic as outlined in Section 9.2 has to include incidents. Apart from their mere detection, the classification plays an important role for InTURN as well. For instance, the road guidance in response to an incident might vary according to its type. First, only cars, heavy load traffic and public transport are taken into account as possible vehicles within network. Pedestrians, bikes or crossing trains are neglected as the focus is on road traffic. Table 9.1 describes types and subtypes of incidents under consideration, together with potential follow-up incidents.

Table 9.1. Traffic incidents under consideration in the context of InTURN. Some can be divided into subtypes with varying characteristics. The listed potential follow-up events might prove valuable for validation of the original events.

type	subtype	descriptive example	possible effects
accident	light	broken down vehicle	slow vehicles, blocked lane
	severe	collision	slow/rescue vehicles, blocked lanes
	critical	severe accident	heavy/rescue vehicles, full block
gauge reduction		smaller lanes due to road works	speed reduction or blocked road
slow vehicles	single	“Sunday driver”	
	multiple	gawker or convoy	
	heavy duty	heavy duty trucks	escorting traffic
blocked road	single lane	road works	slow vehicles
	partial	spilled cargo	
	full	various causes	
escorting traffic		police cars (with warning lights)	slow vehicles
blocked junction	partial	broken down car	blocked road for incoming and outgoing sections
	full	accident	
speed reduction	single lane	planned and maybe announced speed reduction	slow vehicles (compared with other sections)
	partial		
	full		
rescue vehicles		ambulance, possibly ignoring red lights	accidents and slow vehicles
loading / unloading	small / large	unloading/loading, one or multiple lanes blocked	gauge reduction and slow vehicles
stopping bus		bus stopping without a dedicated lane	slow vehicles
road works	single lane	temporary or ongoing road works	slow vehicles
	partial		
	full		
road condition	weather	frost or heavy rain	accidents and slow vehicles
	pollution	spilled cargo	

With respect to these incidents several features can be defined. These are the basis for the design of a sufficient training and test set, which is necessary for the incident-oriented and collaborative incident detection outlined in Sections 9.5.1 and 9.5.2. Part of this design is specifying the details of these features. Also, it has to be assessed if the variance within the features is significant for all types: Maybe some incidents, like “road works” and “blocked roads”, can be combined. The selected features can be divided into 3 groups: spacial, temporal and attribute features.

9.3.1 Spatial Features

These incident features specify spatial aspects with regard to the road networks modelled in 9.2.1. The Table 9.2 lists how these features map to the incidents under consideration.

Locality describes what area of the network is potentially affected. This only involves the type – sections and intersection – independent of the actual network topology or traffic demand:

- intersection*: intersection (fully or partly affected)
- section*: single point within a section
- partial section*: longer stretch of a section
- whole section*: whole length of a section
- anywhere*: anywhere in the network

Location in contrast describes certain places within a particular network where this incident can happen:

- restricted*: only possible in parts of the network
- fixed*: at known locations (e.g., loading ramps)
- anywhere*: anywhere in the network

Breadth of an incident in a section is expressed as number of affected lanes:

- single*: only one line
- multi*: multiple, but not all lanes
- full*: all lanes

Movability of the incident within the road network:

- stationary*: no movement at all
- section*: movement within a section only
- across intersection*: from one section to another
- partial network*: within part(s) of the network
- anywhere*: unrestricted movement

Table 9.2. Spatial features of the incidents outlined in Table 9.1.

type	subtype	Spatial Features			
		locality	location	breadth	movability
accident	light	anywhere	anywhere	single	stationary
	severe			multi	
	critical			full	
gauge reduction		partial section	anywhere	single or multi	stationary
slow vehicles	single	anywhere	anywhere	single	anywhere
	multiple		restricted		
	heavy duty				
blocked road	single lane	partial or whole section	anywhere	single	stationary
	partial			multi	
	full			full	
escorting traffic		anywhere	anywhere	single or multi	anywhere
blocked junction	partial	intersection	anywhere	full	stationary
	full				
speed reduction	single lane	partial or whole section	anywhere	single	stationary
	partial			multi	
	full			full	
rescue vehicles		anywhere	anywhere	single or multi	anywhere anywhere
loading / unloading	small	section	anywhere	single	stationary
	large	partial section	restricted	multi	
stopping bus		anywhere	fixed	single	stationary
road works	single lane	partial or whole section	anywhere	single	stationary
	partial			multi	
	full			full	
road condition	weather pollution	anywhere	anywhere	any	stationary

9.3.2 Temporal Features

The second import class of features deals with the temporal aspects. This is a coarse classification, as especially the duration is dependent on the simulation (time) in question and varying distributions might be necessary. Table 9.3 outlines the temporal features of the relevant incidents.

Duration of the incident within the simulation time:

quick: less than a minute or the TLC cycle time

medium: minutes to maybe half an hour

long: hours to days

permanent: during the whole simulation

Regularity of the times an incident occurs:

periodic: hourly, daily, weekly, ...
special: follows time tables (e.g., public transport)
random: uniquely or irregular

Time when the incident (usually) occurs:

daytime, nighttime: night or day
mornings, afternoon, evenings: time periods of the day (e.g., rush hour)
fixed: fixed time or time period
anytime: no restriction

9.3.3 Attribute Features

The following features go beyond the spatial and temporal aspect. They could potentially be taken into account for the dynamic road guidance. Table 9.3 gives more details regarding the considered incidents.

Predictability denotes whether the time of the incident occurrence is known or can be reasonably predicted:

full: point in time is known, e.g., planned road work
partly: known/assumed probability or “on short notice”
none: unpredictable

Subject involved in the incident:

one: single vehicle
multiple: two or more vehicles, maybe of varying type
road: e.g., in case of road pollution

9.4 Research Objectives

The general concept of the InTURN system is illustrated in Figure 9.4 and helps to formulate related research questions: Each intersection in an urban traffic network is under the control of a component following the Observer/Controller architecture. These components can alter a TLC based on traffic incidents detected by the observer. Components of neighbouring intersections can communicate along the road segments to validate these detections and establish progressive signal systems and/or issue dynamic route guidance (DRG).

Based on this outline of InTURN, overall objectives can be identified:

1. Automated detection of short-term and long-term traffic incidents at the level of individual road segments by supervising intersection controllers that constantly improve the detection performance.

Table 9.3. Temporal and attribute features of the incidents outlined in Table 9.1.

type	subtype	Temporal Features			Attribute Features	
		duration	regularity	daytime	predictability	subject
accident	light	medium	random	any	none	one or multiple
	severe	long				
	critical	persistent				
gauge reduction		persistent	any or special	any	partly	road
slow vehicles	single	medium	random	any	none	one or multiple
	multiple					
	heavy duty					
blocked road	single lane	long	any or special	any	partly	road
	partial					
	full					
escorting traffic		medium	any	any	partly or none	multiple
blocked junction	partial	long	any	any	partly	road
	full					
speed reduction	single lane	long to persistent	any	any	partly	road
	partial					
	full					
rescue vehicles		medium	random	any	none	one or multiple
loading / unloading	small	quick to medium	any	daytime	partly or none	one or multiple
	large					
stopping bus		quick	special	any	yes	one
road works	single lane	long to persistent	special	any	full	road
	partial					
	full					
road condition	weather	medium	random	any	partly	roads
	pollution	to long		daytime	none	road

2. Cooperation of controllers to identify, validate and evaluate traffic incidents beyond the constraint of single road segments.
3. Resilient traffic management through novel, incident-based approaches for self-adaptive and self-organising traffic control.

These objectives correspond to several research challenges for which appropriate techniques have to be developed:

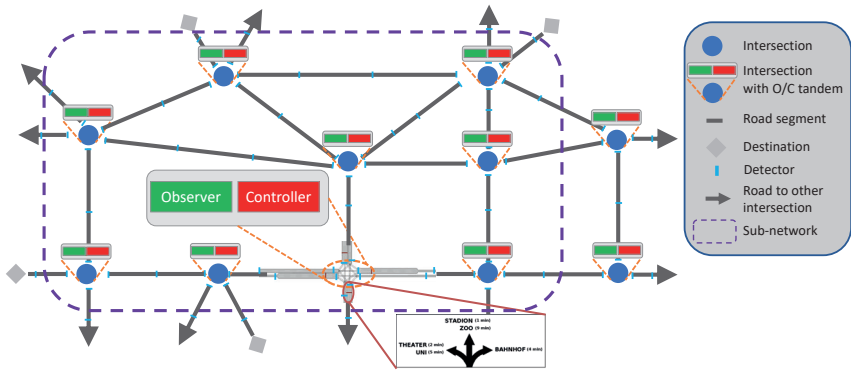


Fig. 9.4. Illustration of the self-adaptive InTURN system. Traffic network intersections are controlled by components, which observe and detect traffic resp. traffic incidents. They control – optionally using cooperation – the TLC and issue route recommendations for drivers.

9.4.0.1 Intersection-oriented incident detection

For the initial detection and classification of traffic incidents a time series analysis of the sensor data will be conducted. Different approaches and their combinations have to be evaluated for an autonomous and reliable classification. The detection scope is defined by the sensor horizon of an intersection.

9.4.0.2 Collaborative incident detection and validation

As drivers traverse the network, neighbouring intersection controllers provide inter-dependent traffic observations. These can be employed to validate and improve the incident detection by correlating expected arrival times according to the known traffic demand.

9.4.0.3 Incident-aware traffic management

Guidance mechanisms have to be developed that consider and communicate the detected incidents in their route recommendations. They could also be used for an autonomous notification of rescue services or for an automatic alteration of speed limits.

9.4.0.4 Capacity-aware traffic management

Route guidance that is solely based on intersection-oriented decision making cannot necessarily accommodate for network-wide traffic streams. Therefore, a region-wide regulation that considers available traffic capacities is required.

9.4.0.5 Evaluation

Finally, a running implementation of the InTURN system is necessary to assess the assumptions made and the final performance using metrics like travel times, numbers of stops, emissions and correctness of incident classifications.

9.5 Research Tasks and Methods

In order to address the research challenges above, several aspects have to be investigated. They partly build upon each other and are outlined below.

9.5.1 Intersection-oriented Incident Detection

The starting point for the initial detection within the responsibility of an intersection controller are lane-based techniques. One example already implemented in OTC is the California algorithm family [5, 6]. However, this is more geared towards highway traffic. These techniques usually rely on thresholds to determine conspicuous traffic conditions between adjacent detectors. Such boundaries are learnt using simulated test data.

Furthermore, novel detection methods based on time series analysis (for instance, clustering) evaluate the normal road usage regarding *vehicles/hour*. For the classification of incident types similarity measures are devised to cluster incidents of similar behaviour and derive prototypes for each. Those are then used to predict the impact on the road segment. The classification is improved during runtime through reinforcement learning, using a-posteriori feedback of incident reports. This is based on the preliminary work of the OTC system with variants of Wilson's Extended Classifier System (XCS) [12], which can be improved by employing other classifiers in an ensemble approach [3].

9.5.2 Collaborative Incident Detection and Validation

To further decrease the false alarm rate (detector readings wrongly classified as incidents) and to detect faulty sensors, the existing information exchange protocols of OTC (as used for establishing a PSS) needs modification: Controllers get ("pull") information from neighbours on traffic running over shared roads and the respective TLC. This facilitates a collaborative validation of stream-based incident classifications and also a collaborative self-assessment of detector plausibility through sensor data comparison. Similar to the previous section, a reinforced reliability estimation of neighbouring intersection controllers – again based on an XCS – is applied for situations *independent* of traffic incidents.

9.5.3 Incident-aware Traffic Management

The OTC system provides a control loop on top of each parametrisable TLC for an intersection. This follows the Observer/Controller design pattern. The goal here is to find methods to employ incident information within OTC for incident-responsive self-adaptation of intersection controllers, PSS and route guidance.

9.5.4 Capacity-aware Traffic Management

The local strategies for altering the signalisation can have network-wide effects. For example, independent changes to favour some traffic flows can lead to congestion problems, when these flows merge “down the road”. To accommodate this, a hierarchical component is introduced to assess the impact of such decisions on the capacity utilisation. It is focused on the traffic demand and uses a simulation-based approach, which employs a parameterisable topology model in Aimsun Next [2]. The goal here is twofold: To what extent are the route recommendation accepted by the drivers? And how is the traffic capacity utilised?

9.5.5 Evaluation

As an extension to the assessment of individual tasks presented so far, in-depth evaluation and analysis of the implemented InTURN system is conducted. This includes a comparison with other state-of-the-art solutions, if available. Based on OTC and the underlying traffic simulations in Aimsun Next, the system performance is assessed using existing, annotated real-world models as in Figure 9.2a (from Hanover and Hamburg, Germany with about 10 – 15 intersections each) as well as newly developed models with up to 100 intersections.

9.6 Summary

At the time of writing, the software migration mentioned in Section 9.3 and the concept of the incident model in Section 9.3 has been addressed.

The further goal of this PhD project is a self-adaptive and self-organising system of intersection controllers that can reliably detect and classify traffic incidents of neighbouring sections. Furthermore, they are capable of cooperation to validate those incidents and improve the performance employing reinforcement learning: The traffic controller can quickly and appropriately alter signalisation and issue route recommendations comprising the traffic disturbances. It can also accommodate for capacity shortages using an additional hierarchical component to reduce congestion effects network-wide. This all can be demonstrated in the context of a traffic simulator with artificial urban road networks and some real-world examples.

Acknowledgements

The project *Incident-aware Resilient Traffic Management for Urban Road Networks (InTURN)* is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), TO-843/5-1. We gratefully acknowledge the financial support in this context.

References

1. Aimsun SLU: Aimsun Next Professional, Version 20. Barcelona, Spain (2020), <http://www.aimsun.com/>
2. Barceló, J., Codina, E., Casas, J., Ferrer, J., García, D.: Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. *J. of Int. and Robotic Sys.* 41(2–3), 173–203 (2005)
3. Dietterich, T.: Ensemble learning. *The handbook of brain theory and neural networks* 2, 110–125 (2002)
4. Müller-Schloer, C., Tomforde, S.: *Organic Computing – Technical Systems for Survival in the Real World*. *Autonomic Systems*, Birkhäuser (October 2017), ISBN: 978-3-319-68476-5
5. Payne, H., Tignor, S.: Freeway incident-detection algorithms based on decision trees with states. *Transp. Res. Rec.* (682) (1978)
6. Payne, H.J.: Freeway incident detection based upon pattern classification. In: *Proc. of IEEE Conf. on Decision and Control*. vol. 14, pp. 688–692. IEEE (1975)
7. Prothmann, H., Tomforde, S., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: *Organic Traffic Control*. In: *Organic Computing – A Paradigm Shift for Complex Systems*, pp. 431 – 446. *Autonomic Systems*, Birkhäuser (2011)
8. Sommer, M., Tomforde, S., Hähner, J.: *An Organic Computing Approach to Resilient Traffic Management*. In: *Autonomic Road Transport Support Systems*, pp. 113 – 130. Birkhäuser, *autonomic systems edn.* (2016)
9. Stein, A., Tomforde, S., Rauh, D., Hähner, J.: Dealing with Unforeseen Situations in the Context of Self-Adaptive Urban Traffic Control: How to Bridge the Gap. In: *Proc. of 13th IEEE Int. Conf. on Autonomic Comp.* pp. 167 – 172 (2016)
10. Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Possibilities and Limitations of Decentralised Traffic Control Systems. In: *Proc. of 2010 IEEE World Cong. on Comp. Int. (IEEE WCCI 2010)*, pp. 3298–3306 (2010)
11. Tomforde, S., Prothmann, H., Rochner, F., Branke, J., Hähner, J., Müller-Schloer, C., Schmeck, H.: Decentralised Progressive Signal Systems for Organic Traffic Control. In: *Proc. of 2nd IEEE Int. Conf. on Self-Adaption and Self-Organisation (SASO'08)*. pp. 413–422 (2008)
12. Wilson, S.W.: Classifier Fitness Based on Accuracy. *Evolutionary Computation* 3(2), 149–175 (1995)

Towards a Self-Aware Prediction of Critical States

Marwin Züfle

Software Engineering Group, University of Würzburg, Germany
marwin.zuefle@uni-wuerzburg.de
Homepage: descartes.tools

Abstract. In the era of digitalisation, more and more systems are being monitored automatically, resulting in huge amounts of data. Yet, most data is currently stored in databases but never analysed further. However, these enormous data sets have the potential to form a basis for many highly practical applications. One of these scenarios is the prediction of critical conditions. The best known discipline in this area is failure prediction, which is the most important step towards predictive maintenance technologies in Industry 4.0. This paper presents the vision of a system model tailored for automatic application to monitoring data for predicting failures and other critical conditions. The system model builds upon the design concept of self-aware computing systems and integrates a meta analysis component for method recommendation. At that time, first experiments have already been carried out and their results show a promising performance.

Keywords: Machine Learning, Failure Prediction, Time Series Forecasting, Method Recommendation, Self-Aware Computing Systems

10.1 Introduction

The handling of changes and critical states is one of the most important tasks for autonomous computing systems [32]. Therefore, such systems implement mechanisms that analyse incoming monitoring data of the system and its environment. Typically, models, thresholds, and utility functions are used to characterise the current state of the system [19]. However, due to digitalisation and increased computing power, these systems can nowadays collect much more information and store it for future analysis. Thus, more advanced data-driven models can be applied to enable the integration of proactive adaptations [19, 32]. The advantage of such proactive systems over typical reactive adaptation-based systems is that the delays in the adaptation process can be eliminated. In addition, in case of critical failures, reactive identification of the current system state is pointless, whereas a proactive system could have predicted this event in advance.

To make a reactive system proactive, the system must have estimates of future states. In most cases, however, a priori modelling of the situations in which a system might reside at runtime is not possible. Consider hard disk drives, for instance. In

the past, a lot of work was put into selecting and collecting meaningful parameters to estimate the current state of health of such hard disk drives [27]. Based on these parameters, experts have derived thresholds indicating whether a hard disk drive is still operating normally or not. Using this expert knowledge at design time, as currently implemented in most hard disk drives, only results in a system that detects current anomalies. However, a more sophisticated system would not only signal current malfunctions, but would also indicate an impending failure in advance. Therefore, the mere integration of expert knowledge at design time is not sufficient for such proactive autonomous computing systems. Thus, prediction methods must be integrated into the system. However, simply adding machine learning predictors is not enough. Instead, the system must perform more extensive data analysis and several challenges must be overcome. First, the target must be clearly defined. Based on the target, meaningful labels for the prediction methods must then be created, on which the quality of the overall prediction depends heavily. Normally, monitoring applications are not error-free, resulting in gaps in the recordings. However, most prediction methods cannot handle missing values, so these gaps must be reconstructed. Next, the system must preprocess the raw data to generate reasonable features. Of these features, only the most relevant can be selected to reduce the time required for model learning and to avoid distortion of the model. Another crucial task is the selection of a suitable prediction method and the optimisation of its hyperparameters. Finally, the system has to update the initial, offline learned model during runtime to adapt to changes in the system and the environment. For this purpose, different strategies can be applied to achieve a trade-off between computing time, required computational resources, and model accuracy.

In this paper, we present the vision of a meta self-aware system model for the prediction of critical states, which addresses all the challenges introduced in the previous paragraph. Regarding the system architecture, we have chosen self-aware computing systems because the design of the explicit learning and reasoning components fits very well with our system model. Nevertheless, the system model can also be transferred to other system architectures, such as autonomous and organic computing systems (cf. Section 10.7).

The remainder of this paper is structured as follows. Section 10.2 contains background information and a nomenclature of the terms used in this paper. Then, we present the research questions posed in this PhD project in Section 10.3. In Section 10.4, we briefly describe the idea of self-aware computing and propose our vision of a meta self-aware system model for predicting critical states, followed by a summary of possible use cases and preliminary results in Section 10.5. Section 10.6 summarises existing related work, while the relation of this work to organic computing is described in Section 10.7. Finally, we conclude the paper in Section 10.8.

10.2 Critical State Prediction

This section provides relevant background information for the prediction of critical conditions that are required for the understanding of this paper. In particular, critical

states can be divided into two types: (I) specific events, such as failures, and (II) critical conditions, which are determined by their behaviour during a time period. Moreover, various common prediction methods are presented.

10.2.1 Failures as Events

The first and most intuitive representation of a critical state is a concrete event. Here, we define such a critical event as follows:

“A critical event is an incident that occurs only at low frequency and its occurrence causes malfunctions of the observed environment.”

Thus, the most common critical event is a failure. However, we do not only consider typical failures in a technological sense, such as outtakes of machines, but also failures in a biological sense. In terms of biology, failures can for instance be found in medicine (e.g., arrhythmia, apoplectic stroke, cardiac infarction) or nature (e.g., extinction of a certain colony or an entire species).

10.2.2 Critical Conditions

The second category of critical states are time spans that show a significantly different behaviour compared to the expected behaviour. Therefore, we define them as follows:

“A critical condition is a future trend that deviates from normal behaviour and therefore implies faults.”

That is, there is no specific point in time that can be considered the critical event. Rather, the entire development must be regarded as critical. Therefore, the future must be predicted, followed by some kind of anomaly detection, which classifies a whole period of time as critical or non-critical. However, this type of critical state is more difficult to predict because both the prediction of the future trend and the classification must operate properly. If the future trend prediction does not provide accurate estimates, the results of the anomaly detection can be arbitrary. An example of such a critical condition would be the anomalous weight development of a bee colony over one year. Here, for instance, the first months could be analysed to predict the remaining development until the end of the year and then classify whether the result is anomalous or not.

10.2.3 Prediction Methods

This section introduces common prediction methods covering a broad range from simple univariate time series forecasting methods to highly complex deep learning models.

10.2.3.1 Time Series Forecasting

The research field of forecasting mainly focuses on univariate, equidistant time series. A univariate, equidistant time series ts with length n is an ordered set of n observations

o , where each observation o_i is mapped to a unique point in time t_i and the temporal difference between the observations Δt is of equal length. On the basis of such time series, common forecasting methods examine the historical data to generate a prediction (typically referred to as a forecast for forecasting methods) for the future. These methods can be either one-step-ahead—i.e., they forecast only one observation—or multi-step-ahead, i.e., they forecast an arbitrary number of observations. Common forecasting methods comprise Autoregressive Integrate Moving Average Model (AR-IMA) [4], Exponential Smoothing State Space Model (ETS) [13], Trigonometric, Box-Cox Transformation, ARMA Errors, Trend and Seasonality Model (TBATS) [8], and Generalized Autoregressive Conditional Heteroscedastic Model (GARCH) [3]. However, according to the “No-Free-Lunch” theorem [33], no method performs best on all data. Therefore, current research in the field of time series forecasting mainly focuses on hybrid methods that combine existing individual methods in an intelligent way to overcome the disadvantages of one method with the advantages of another method.

10.2.3.2 Machine Learning Methods

In contrast to time series forecasting, machine learning methods require features to learn a relationship between the desired outcome (i.e., target or label) and the features. Thus, machine learning models do not directly learn the time dependence between successive observations. Therefore, out-of-sample prediction is not directly possible. To solve this problem, the features can, on the one hand, be forecast using the techniques presented in the section above. Then, these forecasts of the features can be passed to the machine learning methods to predict future observations. In a similar way, the original observations can be lagged and passed as features to the machine learning algorithm. However, when using this technique, only a limited number of predictions can be made, i.e., only as many as the minimum of the applied lags. On the other hand, the labels can be created in such a way that they inherently contain the time component. To come back to failure prediction, the target can be represented as time to failure instead of binary (i.e., failure is present or not). Typical machine learning methods are linear and logistic regression, decision trees, Naive Bayes [24], Support Vector Machine [7], Random Forest [5], and gradient boosting algorithms, such as eXtreme Gradient Boosting (XGBoost) [6].

10.2.3.3 Deep Learning Methods

Similar to the machine learning methods, the deep learning models also require features. However, deep learning models require many more training samples to learn the relationship between the features and the label. In return, these methods can learn more complex relationships and require less feature engineering. Deep learning methods consist of the family of neural networks, especially those with a large number of nodes and multiple layers. The challenge in using deep learning methods is to find a suitable network architecture and hyperparameter setting. There are many different types of neural networks, such as feed-forward neural networks, convolutional neural

networks, and recurrent neural networks, including the currently highly popular long short-term memory neural networks [12]. The advantage of recurrent neural networks over the other types of neural networks is that they can model and learn the time dependence explicitly. Finally, different neural network types can also be combined.

10.3 Research Questions

Starting from the vision of a meta self-aware system model for critical state prediction along with the resulting challenges (cf. Section 10.1), we have posed the following research questions for this PhD project:

RQ1: How do hybrid time series forecasting methods perform compared to state-of-the-art individual time series forecasting methods?

RQ1.1: How can hybrid time series forecasting methods be developed?

RQ1.1: To what extent outperform hybrid time series forecasting methods state-of-the-art individual time series forecasting methods?

RQ2: Which components of the LRA-M loop (cf. Section 10.4.1) must be adapted in what way to integrate the critical state prediction into self-aware computer systems?

RQ3: How can we derive current degradation states of industrial machines only based on standard monitoring data without additional domain knowledge?

RQ4: In which way can we implement a proactive component to predict imminent failure events of technical systems?

RQ4.1: How can we balance the numbers of instances for different failure classes?

RQ4.2: How can we model the time-to-failure?

RQ4.3: Which machine learning models are applicable to critical event prediction?

RQ5: How can we adapt our machine learning methods for critical event prediction at runtime using newly incoming data?

RQ5.1: Which data should we use to re-learn our models?

RQ5.2: Should we only update our models or completely re-learn them?

RQ5.3: When should we trigger such an update strategy?

RQ6: Can the techniques be transferred from the technological domain to the biological domain?

10.4 Vision of a System Model

This section presents the LRA-M loop concept of self-aware computing systems (cf. Section 10.4.1) and the envisaged adjustments for predicting critical states (cf. Section 10.4.2).

10.4.1 Self-Aware Computing Systems

Based on [16], this section presents an introduction to self-aware computing systems. The design of self-aware computing systems focuses on computing systems that use monitoring data to learn models about themselves and the environment. In this way, these systems gain knowledge and use it to reason and act based on these findings. Throughout the entire operation, self-aware systems aim to fulfil higher-level goals. The main concept of self-aware computing systems is the LRA-M loop with its main components Learning, Reasoning, Acting, and Monitoring. This loop is also shown in the top left corner of Figure 10.1. The *self* has several interfaces that are used to monitor itself and the environment, and to receive the higher-level goals as input. Then, it continuously learns a model based on the monitoring data. In this way, it includes both initial offline learning and continuous online re-learning. These learned models, the self itself, and the goals constitute the knowledge base. The reasoning component uses this knowledge base and newly incoming monitoring data to derive a finding that may trigger an action. The self monitors the action and its outcome, which in turn affects the learning and reasoning of the self.

10.4.2 Meta Self-Aware Analysis for Critical State Prediction

In order to integrate proactive analysis for predicting critical states into self-aware computing systems, primarily the learning component must be adapted. Figure 10.1 shows our vision of a meta self-aware analysis in the typical LRA-M loop with the goal of predicting critical states.

10.4.2.1 Preprocessing

Within the learning component, the monitoring data is transferred to the learning procedure of the critical state model. First, the raw data must be processed. The system model contains several methods for this purpose. One method calculates general characteristics directly on the raw input data, such as mean value, variance, skewness, and kurtosis. These features are applicable to almost all scenarios. In addition, more specific characteristics can be calculated for certain areas, such as RR intervals (i.e., the time between two successive heart beats) for electrocardiogram data. Before calculating these features, the preprocessing step can also apply data transformation techniques, such as seasonal and trend decomposition, wavelet transform, and log transformation, or it can detect different phases of the input signal and split it accordingly.

10.4.2.2 Time Series Forecasting

In addition to the feature calculation, the system model can forecast the incoming data to obtain estimates of future developments. This can then be used as an additional feature for the main learning component of the model. Regarding forecasting, we have already developed a novel method called Telescope [2, 36], which divides the

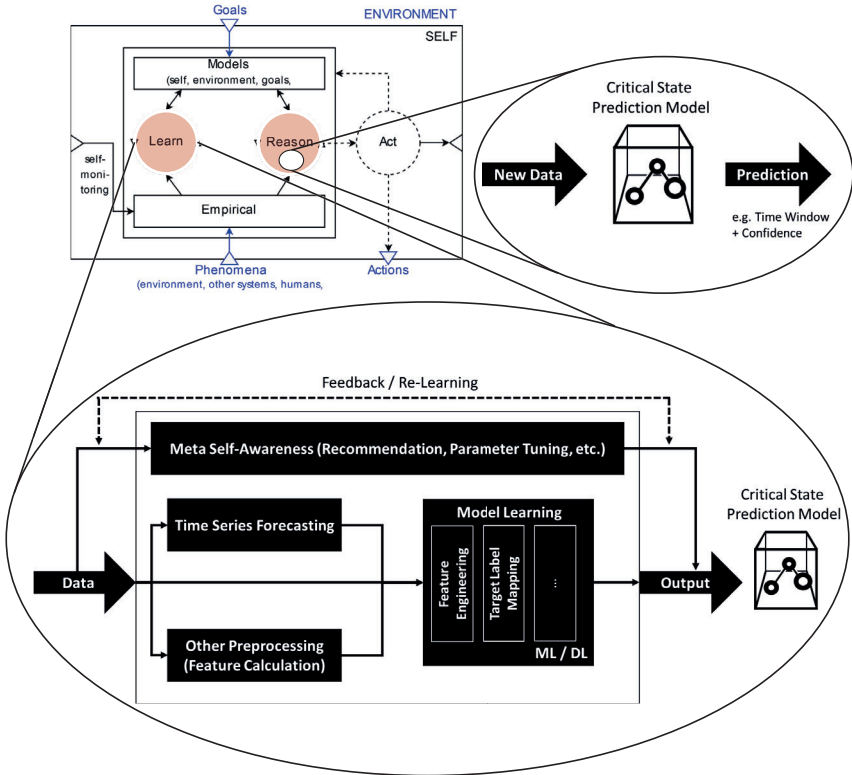


Fig. 10.1. Vision of a meta self-aware system model for critical state prediction.

task into several smaller subtasks, each of which is solved by a different specialised method. That is, the time series is split into trend, season, and remainder components using STL decomposition. For this purpose, however, the frequency of the seasonal pattern must be known. We have developed such a frequency estimation method using spectral analysis (i.e., periodogram). This algorithm examines the frequencies of the periodogram with the highest spectral values and compares them with a list of reasonable frequencies. After decomposing the time series, the seasonal pattern is continued, while an ARIMA model is fitted to the trend component to forecast it separately. In addition, a multi-layer perceptron is trained on aggregated properties of the time series to identify and forecast a potential second seasonal pattern. Finally, an XGBoost model learns the relationship between the forecasts of the multilayer perceptron, ARIMA, and seasonality with the original values of the time series. More precisely, the XGBoost model has to learn the residual component, since this cannot be predicted by a statistical model due to its non-stationarity and noise. Finally, XGBoost provides the forecast of the univariate time series [36]. However, not all

of the forecasting and preprocessing techniques need to be used when applying our system model. Instead, the appropriate parts must be selected by the operator at design time.

10.4.2.3 Model Learning

Afterwards, the raw data is transferred to the model learning component along with the calculated features and the forecasts. Here, all information must be processed to create a meaningful target. The choice of target type depends heavily on the context in which the system model is used. To predict critical developments, the target is a future value of the same time series and therefore, such a problem is designed as a regression task. With regard to critical events, a useful target would be the time to failure or the remaining useful life. This can also be modelled as a continuous regression task or as a discretised classification task. The targets must then be mapped to the respective training instances of the input data. Yet, before the machine learning (ML) or deep learning (DL) model can be learned, a feature engineering process is required. For this purpose, only those features relevant to the respective target are selected and potentially transformed. Finally, the relationship between the selected features and the target is learned using machine learning or artificial intelligence methods and a model is returned containing this knowledge.

10.4.2.4 Meta Self-Awareness

As this process is rather static, a meta self-aware component (sometimes also called “self-reflection”, see [29]) is integrated, which focuses on method recommendations and hyperparameter tuning. Based on time series characteristics, this component can select suitable forecasting and machine learning methods. We have already developed and published these recommendation approaches [1, 35, 37]. The first two approaches calculate several statistical characteristics of the time series in the training data set, such as serial correlation, non-linearity, self-similarity, and chaos, and apply each forecasting method in the recommendation pool to the time series. Based on these results, the best forecasting method can be determined for each time series in the training data set. Then, a random forest classification model is learned for each forecasting method, which estimates whether the respective forecasting method could perform best considering the time series characteristics. However, the classification model does not directly provide the classes, but the class probabilities. Finally, it is proposed to apply the forecasting method with the highest probability of being the best method among all random forest models to the time series under consideration. We have shown that this recommendation approach exceeds the current state of the art in forecasting method recommendation [37]. In contrast, the latter approach is not based on a large training set. Instead, the available data of the time series to be forecast is shortened by the length of the requested horizon. Then, all forecasting methods in the recommendation pool are applied to this shortened time series and their accuracy is assessed based on the retained data. Finally, the forecasting method that performs best on this part is suggested, since it is assumed that the actual horizon

and this artificial horizon are as close as possible, so that the same method should work best on both horizons. This method was used in a forecasting competition, where it significantly outperformed individual methods and also outperformed most other submissions, which is why we were invited to contribute a paper at the conference as one of only four out of 27 competition submissions [35]. However, hyperparameter tuning has not yet been developed.

10.4.2.5 Feedback Loop

To improve the performance of the model, the system model includes a feedback loop that monitors the actual result and compares it with the predicted one. Thus, this component is also responsible for ongoing learning. Based on a pre-defined trigger, this component initiates a re-learning of the model based on the new data received since the last model update. This trigger could be, for instance, simple time spans (e.g., daily, weekly, etc.), deviations between the expected behaviour and the observations (e.g., accuracy below 90%, runtime accuracy below 95% of training accuracy), or a measure for determining a significant concept drift (i.e., the Hoeffding Bound [9]). However, the data used for re-learning the model must also be analysed. On the one hand, the easiest way is to use all available data for re-learning the models. On the other hand, if there is a concept drift in the data so that older data does not properly represent the new incoming data anymore, it might be advantageous to use only the data gathered since the last update. Finally, it should be considered whether the models should be updated (incremental learning) or completely re-learned (re-training). While machine learning models can usually only be re-trained since they do not support online updating, artificial intelligence models can also be updated using the new instances.

10.4.2.6 Reasoning

The application of the derived critical state prediction model is performed in the reasoning component. Here, the new monitoring data arrives, which is then forwarded to the critical state prediction model, including potential preprocessing steps. The model provides the previously defined type of prediction. These results are then forwarded to a planning module to determine whether an adaptation is necessary. However, this planning module is out of the scope of this work. Possible planning approaches can rely on rules (e.g., [18]), models (e.g., [25]), goals (e.g., [17]), or utility functions (e.g., [31]).

10.4.3 Limitations

Although the first results are very promising, the approaches included in the system model still have certain limitations. First, the time series forecasting method we proposed, Telescope, requires seasonal time series with a length of more than two seasonal patterns. If this is not the case, Telescope simply fits an ARIMA model

to forecast the time series. Second, the main part of the modeling requires a large training data set to learn the relationship between the features and the critical states. However, not only does the data set have to be large, especially the set of critical states must be sufficient. Third, if there is a concept drift in the data, the model is updated using the update strategies in the feedback loop, but still the performance of the model decreases until the new pattern is significant enough in the new data.

10.5 Use Cases and First Results

Although there are many more potential use cases for such a meta self-aware system model for the prediction of critical states, we present here only those where we have already applied first versions of our system model or where the application is planned in the near future.

10.5.1 Technology Domain

Typical use cases in the technology sector are machine failure prediction in Industry 4.0 and disk drive failure prediction in cloud computing.

10.5.1.1 Industry 4.0

The concept of Industry 4.0 already includes continuous online monitoring, which provides the necessary database for prediction methods. Furthermore, another goal of Industry 4.0 is the reduction of human intervention. This fits very well with the idea of self-aware computing systems. However, for a production plant to run (semi-)autonomously, imminent machine failures must be known in advance so that countermeasures can be initiated. This is where the critical state prediction component comes into play.

We have already used some components of our system model for the autonomic detection of tool degradation for CNC machines, which showed promising results. We have also written a paper on this topic, which is currently under revision. In that paper, we propose an end-to-end workflow based on machine learning, which is not tailored to the specific machine, but is generally applicable to industrial multi-tool machines. First of all, the workflow resamples the data, because typically, only little training data is available. That is, the original signal is scaled down so that n new signals can be derived from a single raw signal by using only every n -th value. The workflow then segments the raw input data into material processing and tool changing phases using k-means clustering. This step is followed by a kind of smoothing since the cluster labels assigned to the timestamps of the raw data are rather noisy. Next, statistical properties are calculated for each phase. This information is used as input for a hierarchical clustering to map the same processing steps to each other. This allows, for instance, to distinguish drilling phases from milling phases. This is a crucial task since faulty tools can only be detected if they are examined individually.

When analysing an entire manufacturing process, the effect of the faulty tool may vanish due to the other good tools. Therefore, a broad range of features is derived for each phase and used as input for the classification models. Thus, the machine learning models classify each tool either as faulty or good. Based on measurements on a real CNC machine, we evaluated our workflow and achieved an average F1 score of almost 91%.

Furthermore, we have used a first draft of our system model to predict the time to failure of a large-scale press. Here, we achieved a highly accurate classification and are currently planning a further publication.

10.5.1.2 Cloud Computing

Another relevant area is cloud computing. According to Backblaze, hard disk drives (the engine of cloud computing) have a comparatively low annual failure rate of only about 2% [15]. However, with large cloud providers running several thousand hard disk drives in parallel, this leads to daily hard disk drive failures. Still, these cloud providers must provide fast and reliable services to end users. Therefore, cloud providers must identify failing disk drives in advance based on monitoring data. To this end, we have already published a paper based on components of our system model that predicts the remaining useful life of hard disk drives with an F1 score of approximately 98% [38].

Here, we used the so-called Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) monitoring data of hard disk drives. These data include various internal parameters and operations, such as the head flying height, spin-up time, and drive calibration retry count. Although the data set covers many more S.M.A.R.T. features for each instance, we have included only the 25 most relevant ones in our experiments. Besides these monitoring data, the data set contains a flag for each instance, which indicates whether the respective hard disk drive is currently faulty or not.

However, since we wanted to predict upcoming failures, these labels had to be changed. First, we calculated the time to the next failure (TTF) for each hard disk drive. If the hard disk drive did not fail during the measurement period, we set the TTF to infinity. For a binary evaluation of whether the hard disk drive will fail within a week or not, the labels of each instance with a TTF of no more than 168 hours were set to 1, all other instances received label 0. As this led to a highly imbalanced data set, we applied two different oversampling strategies, namely the Enhanced Structure Preserving Oversampling (ESPO) and the Synthetic Minority Oversampling Technique (SMOTE). Finally, we learned a random forest classification model with 100 decision trees for each setting, i.e., unmodified (no oversampling), ESPO oversampling and SMOTE oversampling. The results showed that ESPO delivered the best results with an average F1 score of 95.93%, although it improved the classification only slightly compared to the unmodified version.

The next step was not only to determine whether or not the hard disk drive would fail within a week, but also to find out more precisely when the hard disk drive would fail. Therefore, we defined a set of relevant TTF classes, each representing a different failure time span: 0, (0,1], (1,2], (2,5], (5,12], (12,24], (24,48], (48,72],

(72,96], (96,120], (120,144], (144,168], (168,∞). Thus, each of these labels represents the interval of the TTF in hours, while the class label (168,∞) indicates that there will be no failure within the next week. Due to this increased complexity, we have trained the random forest model using 500 decision trees. Table 10.1 shows the resulting confusion matrix with the predicted labels (Pr) shown on the columns and the actually observed labels (Ob) on the rows. For the sake of simplicity and readability, we refer to each of the TTF classes only by its upper limit, e.g., we refer to label 48 instead of label (24,48]. The correct predictions are shown on the diagonal and marked in green. Although this approach predicts the TTF much more fine-grained than the binary approaches, this approach achieves a micro F1 score of even 97.63%.

Finally, we have also downscaled the multi-class classification approach to the same two classes used for binary classification. That is, we have combined all classes except ∞ into one large class, i.e., the class that indicates an imminent failure within the next 168 hours. Evaluating the multi-class classification model on this binary basis yields an even better F1 score of 98.02%. Thus, we have shown that our multi-class classification method exceeds the binary approaches not only in terms of more detailed predictions, but also for the binary case.

Table 10.1. The confusion matrix for the multi-class classification approach. The rows show the actually observed (Ob) TTF classes, while the columns present the predicted (Pr) ones. The value in each cell illustrates the number of instances predicted for that particular set of observed and predicted class labels. The green colour indicates the correctly predicted instances. The second cell from the left in the third row, for example, shows a single instance that is predicted as “a failure will occur within the next hour”, while the failure actually occurred in the time window of one to two hours after the measurement. The confusion matrix is taken from our earlier work [38].

Pr \ Ob	0	1	2	5	12	24	48	72	96	120	144	168	∞
0	67	0	0	0	0	0	0	0	0	0	0	0	2
1	0	35	0	0	4	2	9	0	0	0	0	0	0
2	0	1	12	0	0	2	0	0	0	0	0	0	2
5	0	0	0	52	24	17	1	0	0	0	0	0	2
12	0	0	0	0	185	14	2	0	0	0	0	0	8
24	0	0	0	0	17	339	21	0	0	0	0	0	12
48	0	0	0	0	0	6	773	10	0	0	0	0	18
72	0	0	0	0	0	0	12	740	22	0	0	0	24
96	0	0	0	0	0	0	0	6	768	15	0	0	24
120	0	0	0	0	0	0	0	0	14	752	6	0	24
144	0	0	0	0	0	0	0	0	0	20	762	8	29
168	0	0	1	0	0	0	0	0	0	0	16	729	66
∞	0	0	1	1	0	1	0	1	1	0	1	5	14143

10.5.2 Biology Domain

Apart from the technology domain, critical states also occur in biology. Due to the age of digitalisation, more and more data is monitored and stored online in this area as well. One use case of our system model in biology is severe heart infarctions. In clinics, it happens regularly that patients suffer a severe heart infarction. If this happens outside the intensive care unit, these patients often die or suffer permanent damage. For this reason, we work together with the University Hospital of Würzburg to predict such severe heart infarctions. Here, a 30-second to 2-minute lead and an alarm that reaches doctors and nurses would already help saving many lives. Another biological application is the mortality of insects. Especially the prediction of the development of bee colonies with regard to their population size. Since most agricultural and wild plants are pollinated by bees, they are very important for our crops and biodiversity. Studies have already shown that the biomass of flying insects in protected areas has been reduced by an average of 2.8% per year over the last 27 years [22]. We therefore work together with two bee institutes that have already implemented more than 300 bee colony scales all over Germany to monitor the weight development over the years. On the basis of this data, we want to use our system model to predict critical developments at an early stage so that the beekeeper can take timely countermeasures to keep his beehive healthy.

10.6 Related Work

There already exist several approaches to predict failures for specific applications, such as financial failures [28, 34], bearing failures [11, 23], or product failures [10, 21]. For a general overview of methods for failure prediction, please see the overview of Salfner et al. [26]. However, these methods only focus on failure events and are only applicable to their particular use case. Although it is indisputable that one approach cannot cover several application cases simultaneously, the literature lacks a system model that is integrated into an autonomous computing system architecture and provides functionality to cover this wide variety of use cases.

With regard to autonomous computing systems, there are several other design concepts besides self-aware computing systems. One of them is autonomic computing [14], which was developed in the area of large IT infrastructures. The authors pointed out that the management overhead of such computing systems is increasing more and more and goes far beyond simple administration. In addition, the systems are becoming increasingly interconnected, which massively increases the complexity of handling these systems. For this reason, the authors present an automated backbone component that enables self-management of the computing systems. This includes self-configuration, self-optimisation, self-healing and self-protection. As a result, autonomic computing systems support the administrators and operators by taking over these tasks. Today, autonomic computing is often used as a summary of all kinds of autonomous computing systems. Another design concept is organic computing [30]. Organic computing systems typically aim to shift traditional design time

decisions to runtime [20]. Therefore, these systems must be able to make autonomous decisions with little or no human intervention. Important properties for such a system are self-healing, self-protection, self-stabilisation, self-improvement, and self-explanation [30]. The typical structure of organic computer systems comprises several layers. Layer 0 is the system under observation itself. Above this follows layer 1, which performs the online parameter selection and applies the algorithms. Layer 2 is responsible for offline learning, including parameter optimisation and the generation of new rules for adaptation to new conditions.

10.7 Relation to Organic Computing

To put the vision of a meta self-aware system model for the prediction of critical states into the context of organic computing, this section describes the connection between them. A typical definition of an organic computing system is “*a computer system that acts without or with only limited manual intervention. It thereby achieves and maintains a certain performance or utility even in time-variant environments and distributed situations. In response to the dynamics, it adapts and improves behaviour over time and interacts with other systems to achieve an individual and/or system-wide goal*” [30]. Parts of this definition are similar to the one of self-aware computing systems (cf. Section 10.4.1) since both paradigms aim to minimise human intervention and thus automatise computing systems. This also applies to the proposed system model, since the operator only has to select the necessary components of the system model at the beginning and afterwards, the system learns and optimises by itself by means of the implemented feedback loop. For this reason, the proposed system model also fulfils the last requirement of self-adaptation and self-improvement over time. By analysing the predictions and comparing them with the actual conditions, the system is able to re-learn and thus optimise with the goal of predicting imminent critical states in advance. In addition, the system model is also intended to interact with other systems. For instance, a system model for maintenance planning would be required to fulfil a more system-wide goal, namely the minimisation of the total cost of machine downtime and maintenance operations. Finally, the system model preserves its utility in time-variant environments and distributed situations, as it is able to adapt to unforeseen situations using the environment observation interfaces in conjunction with the feedback loop.

To place the system model into the three-layer architecture of organic computing systems, the observed environment (e.g., industrial machine, hard disk drive, bee colony) can be considered as layer 0. As layer 1 is responsible for the parameter selection and application of the algorithms, it refers to the initial generation and application of the critical state prediction model. Lastly, the feedback loop, which offers the possibility to optimise to current conditions or to adapt to new situations, can be compared to layer 2 of organic computer systems.

10.8 Conclusion

In this paper, we presented our vision of a system model for the prediction of critical states based on the concept of self-aware computing systems. For this purpose, the learning component of the LRA-M loop needs to be adapted to generate a critical state prediction model based on the input data and some preferences defined by the operator at design time. First, an initial critical state prediction model is learned offline, and while the system is running and collecting further monitoring data, the model receives feedback on its predictions and possible changes in the environment. Based on this information, the model is continuously re-learned or updated. The model is then used in the reasoning component to support the planning module in making further decisions. However, the planning module is not part of this work. Up to this point, we have already developed some of the components, while other components and research questions still need to be answered. The first results of the existing components are already very promising, as the predictions of the critical state are highly accurate.

References

1. Bauer, A., Züfle, M., et al.: An Automated Forecasting Framework based on Method Recommendation for Seasonal Time Series. In: Proceedings of the 11th ACM/SPEC International Conference on Performance Engineering (ICPE 2020). ACM, New York, NY, USA (April 2020)
2. Bauer, A., Züfle, M., et al.: Telescope: An Automatic Feature Extraction and Transformation Approach for Time Series Forecasting on a Level-Playing Field. In: Proceedings of the 36th International Conference on Data Engineering (ICDE) (April 2020)
3. Bollerslev, T.: Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics* 31(3), 307–327 (1986)
4. Box, G.E., Pierce, D.A.: Distribution of Residual Autocorrelations in Autoregressive-integrated Moving Average Time Series Models. *Journal of the American Statistical Association* 65(332), 1509–1526 (1970)
5. Breiman, L.: Random Forests. *Machine Learning* 45(1), 5–32 (2001)
6. Chen, T., Guestrin, C.: Xgboost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
7. Cortes, C., Vapnik, V.: Support-vector Networks. *Machine Learning* 20(3), 273–297 (1995)
8. De Livera, A.M., Hyndman, R.J., Snyder, R.D.: Forecasting Time Series with Complex Seasonal Patterns using Exponential Smoothing. *Journal of the American Statistical Association* 106(496), 1513–1527 (2011)
9. Domingos, P., Hulten, G.: Mining High-Speed Data Streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 71–80 (2000)
10. Evans, A., Wiederhorn, S.: Proof Testing of Ceramic Materials—An Analytical Basis for Failure Prediction. *International Journal of Fracture* 10(3), 379–392 (1974)
11. Herp, J., Ramezani, M.H., et al.: Bayesian State Prediction of Wind Turbine Bearing Failure. *Renewable Energy* 116, 164–172 (2018)

12. Hochreiter, S., Schmidhuber, J.: Long Short-term Memory. *Neural Computation* 9(8), 1735–1780 (1997)
13. Hyndman, R., Koehler, A.B., et al.: *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Science & Business Media (2008)
14. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. *Computer* 36(1), 41–50 (2003)
15. Klein, A. (Backblaze): *Backblaze Hard Drive Stats for 2019* (2020), <https://www.backblaze.com/blog/hard-drive-stats-for-2019/>
16. Kounev, S., Lewis, P., et al.: The Notion of Self-Aware Computing. In: Kounev, S., Kephart, J.O., et al. (eds.) *Self-Aware Computing Systems*. Springer Verlag, Berlin Heidelberg, Germany (2017)
17. Kramer, J., Magee, J.: Self-Managed Systems: An Architectural Challenge. In: *Proceedings of the Future of Software Engineering (FOSE '07)*. pp. 259–268 (2007)
18. Krupitzer, C., Drechsel, G., et al.: Using Spreadsheet-defined Rules for Reasoning in Self-adaptive Systems. In: *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops*. pp. 289–294 (2018)
19. Krupitzer, C., Roth, F.M., et al.: A Survey on Engineering Approaches for Self-adaptive Systems. *Pervasive and Mobile Computing* 17, 184–206 (2015)
20. Krupitzer, C., Tomforde, S.: The Organic Computing Doctoral Dissertation Colloquium: Status and Overview in 2019. In: *INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik—Informatik für Gesellschaft (Workshop-Beiträge)*. Gesellschaft für Informatik eV (2019)
21. Ku, J.H.: A Study on the Machine Learning Model for Product Faulty Prediction in Internet of Things Environment. *Journal of Convergence for Information Technology* 7(1), 55–60 (2017)
22. Lamb, E.G., Hallmann, C.A., et al.: More than 75 Percent Decline over 27 Years in Total Flying Insect Biomass in Protected Areas. *PLoS ONE* 12(10), e0185809 (2017)
23. Lee, J., Wu, F., et al.: Prognostics and Health Management Design for Rotary Machinery Systems—Reviews, Methodology and Applications. *Mechanical Systems and Signal Processing* 42(1-2), 314–334 (2014)
24. Maron, M.E.: Automatic Indexing: An Experimental Inquiry. *Journal of the ACM (JACM)* 8(3), 404–417 (1961)
25. Pffannemüller, M., Krupitzer, C., et al.: A Dynamic Software Product Line Approach for Adaptation Planning in Autonomic Computing Systems. In: *Proceedings of the IEEE International Conference on Autonomic Computing (ICAC)*. pp. 247–254 (2017)
26. Salfner, F., Lenk, M., Malek, M.: A Survey of Online Failure Prediction Methods. *ACM Computing Surveys (CSUR)* 42(3), 1–42 (2010)
27. Seagate Product Marketing: *Get S.M.A.R.T. for Reliability*. Tech. rep., Technical report, Seagate Technology Paper (1999)
28. Tam, K.Y., Kiang, M.Y.: Managerial Applications of Neural Networks: The Case of Bank Failure Predictions. *Management Science* 38(7), 926–947 (1992)
29. Tomforde, S., Hähner, J., von Mammen, S., Gruhl, C., Sick, B., Geihs, K.: "know thyself" - computational self-reflection in intelligent technical systems. In: *Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2014, London, United Kingdom, September 8-12, 2014*. pp. 150–159 (2014)
30. Tomforde, S., Sick, B., Müller-Schloer, C.: *Organic Computing in the Spotlight*. arXiv preprint arXiv:1701.08125 (2017)
31. Vansyckel, S., Schäfer, D., et al.: Configuration Management for Proactive Adaptation in Pervasive Environments. In: *Proceedings of the IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*. pp. 131–140 (2013)

32. Weyns, D.: Software Engineering of Self-adaptive Systems: An Organised Tour and Future Challenges. Chapter in Handbook of Software Engineering (2017)
33. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1(1), 67–82 (1997)
34. Xu, X., Wang, Y.: Financial Failure Prediction using Efficiency as a Predictor. *Expert Systems with Applications* 36(1), 366–373 (2009)
35. Züfle, M., Kounev, S.: A Framework for Time Series Preprocessing and History-based Forecasting Method Recommendation. In: *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems*. vol. 21, pp. 141–144. IEEE (2020)
36. Züfle, M., Bauer, A., et al.: Telescope: A Hybrid Forecast Method for Univariate Time Series. In: *Proceedings of the International Work-conference on Time Series (ITISE 2017)* (September 2017)
37. Züfle, M., Bauer, A., et al.: Autonomic Forecasting Method Selection: Examination and Ways Ahead. In: *Proceedings of the 16th IEEE International Conference on Autonomic Computing (ICAC)*. IEEE (June 2019)
38. Züfle, M., Krupitzer, C., et al.: To Fail or Not to Fail: Predicting Hard Disk Drive Failure Time Windows. In: *International Conference on Measurement, Modelling and Evaluation of Computing Systems*. pp. 19–36. Springer (2020)

Self-adaptation using discriminative, dynamic models for activity recognition

Autonomous Theft Detection based on Generative Activity Recognition Models and Novelty Detection

Martin Jänicke¹, Vitor Fortes Rey², Bernhard Sick¹, Sven Tomforde³, Paul Lukowicz²

¹ University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany,
{mjaenicke, bsick}@uni-kassel.de,
<https://www.ies-research.de>

² German Research Center for Artificial Intelligence, Tripstadter Straße 122,
67663 Kaiserslautern, Germany,
vitorrey@gmail.com, paul.lukowicz@dfki.de,
<https://www.dfki.de>

³ University of Kiel, Hermann-Rodewald-Straße 3,
24118 Kiel, Germany,
st@informatik.uni-kiel.de,
<https://www.ins.informatik.uni-kiel.de>

Abstract. Personal devices such as smart phones are increasingly utilized in everyday life. Frequently, activity recognition is performed on these devices to estimate the current user status and trigger automated actions according to the user's needs. In this article, we focus on a novel combination of modeling sensor data and adapting classification systems: We analyze acceleration time series by means of HMMs and use the characteristics as features for a recognition system. That recognition system consists of a classifier that can be extended during run-time with additional sensors. Using the data from the OPPORTUNITY benchmark dataset, we were able to show improvements in the extended classification system, which was trained fully autonomously. For the evaluation we trained and evaluated 288,540 classifiers and showed representative results using representative measures.

Keywords: Hidden Markov Models, Organic Computing, Timeseries classification

11.1 Introduction

Ubiquitous activity and context recognition (e.g., [1, 14, 17]) is a well established research field that aims to translate the information provided by simple sensors into high level knowledge about human activities and the situation in the environment. Concrete examples range from using workers jacket with embedded sensors to recognize individual steps of a maintenance task [53] through the use of home infrastructure

to monitor the behavior of people with cognitive impairments [40] to various applications that leverage the sensors of a smart phone to analyze the user's physical activity [31].

Motivated by the fact that our environment is richly equipped with sensors, we follow an approach where we leverage as much of the provided measurements as possible, i.e., we start with a subset of all available sensors and add further input sources at runtime. This approach is also known as “Opportunistic Activity Recognition (OAR)” and was first introduced in the ubiquitous computing community [47]. Given the sketched scenario, the overall idea is to create *Self-improving* systems, that is, systems that improve their performance with new input sources. Given the same techniques, it is also possible to realize a form of *Self-healing*, where the system performance first drops (e.g., due to sensor failure), and is afterwards regained (or even exceeded) by selecting a replacement-sensor from the environment. Interpreting *Self-integration* as a process where a system integrates another source of information autonomously, shows the third property relevant for our work. These examples show, how such systems realize Self-*properties from the field of Organic Computing (OC) [42].

Sensor data is usually temporal in its nature, i.e., it consists of a time series of sensor readings. These are preprocessed and afterwards handled by some kind of Machine Learning (ML)-algorithm. Two principal approaches can be pursued for modeling. One are static approaches, where temporal information is not taken into account. Methods where features are extracted from sliding windows and processed afterwards fall within this category. Cohesion between successive data is completely neglected, these features can be shuffled in order and the results stay the same. They can be seen as „classical“ approaches. The second are dynamic approaches, modeling/capturing temporal information, e.g., by means of internal storage mechanisms or via feedback mechanisms. Both cases can still be used as starting points for this work, as we research the extension of the input space during runtime. Studies that compare both kinds of methods are available (cf., e.g., [13, 23]) and discuss broadly which approaches have been followed successfully. We investigate a novel method that comes from the second category, using timely information by means of a model combination.

The remainder of this article is structured as follows. In Section 11.2, this work is first put into context among other adaptation practices and narrowed down to the specific field of OAR. Afterwards, Section 11.3 describes the modeling technique we apply on the data, the model creation and the necessary label inference for the adaptation step. This is accompanied by Section 11.4, which presents first results we found after investigating the well known OPPORTUNITY benchmark dataset [48]. Finally, Section 11.5 summarizes the paper and gives an outlook to further research activities.

11.2 Related Work

In the ubiquitous computing community there has long been broad consensus that context recognition systems need to become more flexible and adaptive. However, the

bulk of work performed so far concentrated on interoperability and service discovery as well as on the ability to adapt to changes in sensor properties. The former includes a variety of middleware concepts [6, 37] and systems as well as different sensor self-description methodologies [28, 50, 51]. Examples of the latter include: 1. An approach pursuing the choice of sensor combinations that are tolerant to different body placements (e.g., [33, 39]). Along these lines different sensor fusion approaches were exploited to minimize the impact of changes in sensor placement (e.g., [4]). 2. With the wide-spread adoption of smartphones, different groups have investigated AR methods that utilize smartphone sensors and are tolerant of the smartphone placement (e.g., [18]). 3. On an abstract level there were attempts to treat changes in sensor placement mainly as leading to shifts in the feature distributions. They then attempt to track those shifts at run-time, (e.g., [11]) in unsupervised adaptive classifiers that calibrate themselves using expectation maximization.

So far, there has been little work explicitly devoted to the use of new, previously unknown sensors. In [29], the authors show how a new sensor can be trained using existing ones so that it can “jump in” should one sensor fail. An information theoretic approach to dynamically selecting optimal sensor ensembles including sensors appearing during run-time has been proposed in [12]. However, it assumes that the new sensors are already “trained”. In [10], a methodology is presented that uses sporadic interactions with primitive sensors together with behavioral assumptions to confer AR capabilities to a newly discovered sensor. Related to the question of using previously unseen sensors without additional training data is general research on minimizing the amount of training in AR systems. In addition to more general work in semi-supervised learning, active learning, and transfer learning described below there has been some very interesting work in leveraging online resources for training AR systems. For example, [32] proposes to use social network connections to identify users whose behavior is similar enough to facilitate the sharing of pre-trained classification models and labeled training data. Alternatively, activity models to be used for a recognition system are mined from online information sources [54].

More concrete towards our application, the field of OAR was introduced [30, 36, 46, 47]. Specific work had a strong focus on robustness and adaptation of wearable AR systems to changes in sensor properties. This included automatic calibration of acceleration sensors [35], the ability of the system to recognize where on the body a sensor is located [26] and how it is oriented [27], as well as methods for dynamic feature selection as means of compensating device shift [25].

The research field from Organic Computing helps in formalizing the goal of autonomously adapting sensor systems. In our understanding, the transfer of Self-* properties (Self-organization, Self-optimization, etc.) from design time to run-time is one of the main research fields of OC [43, 55]. The concepts of Self-organization and Self-healing were addressed in the PRISMATICA project. With the goal of enhancing safety in public places, the surveillance took place with smart camera systems that can adjust several degrees of freedom (e.g., their viewing angles) to cover certain areas efficiently [19]. The system is able to detect and account for camera failures or message loss during communication. Similar to our proposal, a highly varying input space (different viewing angles) was examined, but the input space was not extended

during run-time. Attempts to implement reconfigurations mainly cover systems with multiple entities, such as many-core CPUs. Such computing entities are monitored in order to shift load between them depending on certain criteria, considering, e.g., thermal stress (in the case of CPU cores [16]) or reliability (e.g., [52]).

Another example of Self-* property realization was investigated in the “Digital On-demand Computing Organism for Real-time Systems” project (DodOrg) [7]. The work focused on the term “on-demand”, with the goal of achieving a high responsiveness to changes in the environment or the system itself. The issues of Self-protection and robustness with regard to malformed or compromised input data do not lie within our objectives. Maehle et. al [20] conducted a research project on robots with Self-reconfiguration properties. Each leg of an insect-like-robot supplies input information and is able to communicate with each other leg. The detection of malfunctioning legs represents one change in the input space. Detection is possible by a continuous monitoring process, whereas algorithms for handling are based on ideas from the field of swarm intelligence. However, the adaptation by reconfiguration did not include the extension of the input space.

Other approaches focus on modeling uncertainty in all contact points a system has with its environment: input data, output data, or acquired knowledge. Another issue is trustworthiness as a measure of user confidence [9]. Such approaches evaluate the uncertainty of information within the system and are able to regulate their performance depending on the approximated level of uncertainty. An architecture related to the one we propose in this article was developed in a project that focused on the development of a generic middleware that incorporates multiple layers of a system [49]. Those layers were able to communicate with each other to achieve a system wide self-organization. [41] shows how systems may react on observed changes in their input space, e.g., by unsupervised search for new clusters in an input space with fixed dimensions.

Preliminary work in the field of reasearch this paper lies in was mainly performed by Bannach [2, 3]. The author investigated adaptation techniques from lower to higher dimensional input spaces during runtime by extending decision trees and proposing measures to estimate gain and risk for such adaptation-steps. The biggest restriction of the work was the limited dimensionality of the input space. Following those publications was another preliminary work [24], in which the same adaptation datasets were investigated, however with totally different techniques, i.e., based on Gaussian Mixture Models (GMMs). In addition, the problem of parameter complexity was investigated in that paper. These works are the foundation for this publication. For reasons of comparability, the same OPPORTUNITY-dataset as before is used [48], but the feature modeling is totally different: In this work, the goal was to model accelerometer-data and it’s timely behaviour using Hidden Markov Models (HMMs) instead of classic features (e.g., mean and variance of sliding windows of data).

An interesting approach from Tsuda et al. [56] focuses on the usage of a generative HMM to work in conjunction with a discriminative Support Vector Machine (SVM). This is since then known as creating a *TOP-kernel* for SVMs and is one source of inspiration for the method investigated in this paper.

11.3 Method

In our approach we model temporal behavior by means of HMMs, which are used to create features for further processing.

The input data we currently investigate comes from several accelerometer sensors that were mounted to limbs and torso of subjects. In our case, the extension of the system, i.e. the self-adaptation of a classifier, is equivalent to adding k sources to the input space with d input dimensions (resulting in a $d + k$ dimensional input space). Please note that we use the terms “sensor” and “feature” synonymously. This is motivated by the fact that with one additional sensor (e.g. a temperature sensor) and just one extracted feature (e.g. the mean temperature over the last four seconds), it is an obvious one-on-one relation. Secondly, the distinction between sensor and feature is transparent to the classifier, so that we would only emphasize the difference if necessary.

HMMs are used to model stochastic processes, i.e., processes, that have a specific state for each point in time. They were first introduced by Baum et al. [5] and model time discrete processes, where states can change in every timestep with a certain probability (*transition probability*). Successful applications involve, e.g., natural language processing [22], handwriting recognition [38] or the analysis of biological sequences [15]. Details about training and construction of these models can be found in, e.g., [44]. Here, only a brief overview, necessary to understand the overall approach, is given. Given a discrete number of possible states, the goal of the training process is to find best estimates for transition probabilities between these states (stored in a so called *transition matrix*) and output probabilities for specific observations (so called *emissions*). Please note, that the current state of the HMM can not be seen, it can just be estimated via the emissions.

Modeling data by means of a HMM and further processing it with a different ML-algorithms was first introduced as the TOP-kernel for SVMs by Tsuda et al. [56]. However, when investigating that approach on Activity Recognition (AR) data, we found two major issues: 1.) It is not directly applicable to multi-class problems and 2.) An adaptation from univariate to multivariate data is not as straight forward as one might think. While the first problem can be overcome as usual, when going from binary to multi-class problems (to create either several one-against-all or several one-against-one-classifiers), the second overwhelmed us with the curse of dimensionality. The outcome was, that the TOP-kernel lead to heavy overfitting, i.e., an acceptable low training error, but a very bad generalization. Another drawback was, that even if the SVM would have performed reasonably well, so far no adaptation to new input dimensions is available, it boiled down to the (to our knowledge unsolved) question: How can we extend a SVM to leverage new input sources? That is why we switched our focus to still use a generative HMM to capture timely processes in AR data, however, we switched the method afterwards to something more capable of adapting to new sensors.

The conjunction we used for this work is sketched in Figure 11.1. Given the d -dimensional training time series \mathbf{X}_{Tr} , a generative HMM (denoted by \mathbf{G}) is trained, and afterwards used to process the training set, generating specific features \mathbf{T}_{Tr} :

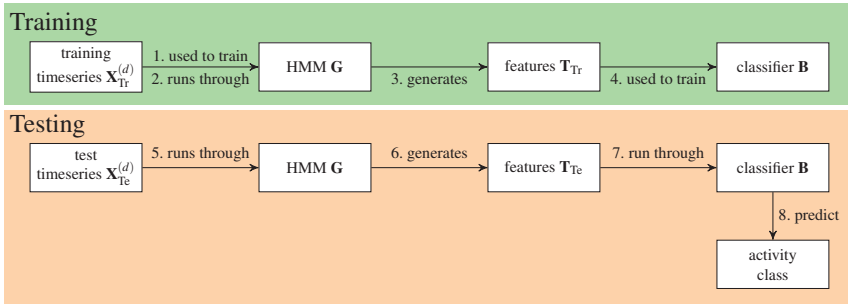


Fig. 11.1. Conjunction of generative data preparation and latter classification.

The characteristics of the HMM, i.e. transition probabilities and output probabilities are the values that form the feature-vectors. These features are then used to train a concrete classifier **B**. In our case, two paradigms are investigated: a Classifier based on Gaussian Mixture Models (CMM) and a decision tree (DT). The former has been chosen due to the marginalization property of the paradigm (cf., e.g., [21]), which is necessary for the input-space-extension, while the second is a paradigm that has been proven to work on the input-space-extension of AR systems (cf., e.g., [2]). For testing purposes, the just created model **G** is used to create features **T_{Te}** from the test-data, which are then run through the trained classifier **B**. The class predictions are then used for evaluation.

In the adaptation case, the adaptation data is used to train a HMM fully unsupervised. The adaptation process is visualized in Figure 11.2. Given the observations

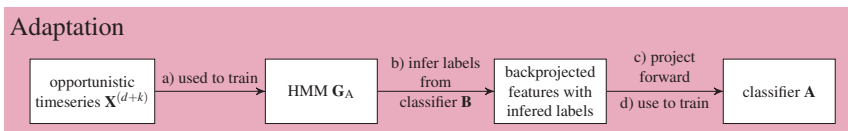


Fig. 11.2. Overall adaption process, from opportunistic data in $d + k$ dimensions, that extends the original d -dimensional input space, to the adapted classifier **A**.

from the d input dimensions of the training time series and the additional k dimensions of opportunistic data, a HMM **G_A** is trained fully unsupervised. The resulting features are backprojected into the d dimensional input space and labels are inferred. In the case of a CMM classifier, the backprojected data is labeled by the original classifier (*base* classifier **B**), in the case of a DT, the labeling probabilities and similarities in the higher dimensional space are used by means of a semi-supervised inspired algorithm, where the original classifications from the training data **T_{Tr}** are propagated using the similarities in the $d + k$ -dimensional input space. The process is described in more

detail in [45]. Finally, the labeled data is forward projected into the $d + k$ dimensional input space, where the *adapted* classifier \mathbf{A} can be trained fully supervised.

The evaluation then happens as described previously, with the difference of issuing predictions from the adapted classifier \mathbf{A} , after it processed $\mathbf{X}^{(d+k)}$.

11.4 Evaluation

For the evaluation of our approach we relied on the OPPORTUNITY AR-dataset [48] as in preliminary works by Bannach [2, 3]. We chose to evaluate the version with five basic activities: lie, sit, stand, walk, null. These activities are also quite commonly investigated in the literature and are expected to be well captured by the HMM-based approach. The investigated data was recorded with 17 different sensors, which were distributed over arms, torso and legs. Overall, data from four subjects with five repetitive sessions per subject. Evaluations took place via leave-one-session-out-cross validations.

For our adaptation scenarios, we chose to split the training dataset (four sessions) in halves, so that we have 50% of the data for actual training and 50% for the adaptation process, to refine the model. To cover different evaluation scenarios, the test-dataset (of dimensionality $d + k$) was either backprojected to the d dimensions of the original system, or, it was left untouched to evaluate the adapted model. We considered the F_1 -score as evaluation measure, as it contains statements about precision and specificity of our approach and thus, more information than, e.g., the classification accuracy.

The features we chose are actually descriptive properties of the HMM, that was trained unsupervisedly on the data. We chose six states under the assumption that the observed is normally distributed. However, we chose to restrict the forms of the covariance matrices to spherical/isotropic (cf., e.g., [8]). This was due to the problem, that otherwise (i.e., with the assumption of random, normally distributed data), more degrees of freedom than available samples would have needed estimation, thus it helped in bringing down the complexity. In the end, we just considered the transition and output probabilities for our experiments. Besides the cross-validation, we investigated all incremental combinations of sensors, i.e., going from one sensor to two, from two to three, and so on. We aimed at testing each combination of sensors as starting set and extending it by one, but the number of possibilities grows exponentially (2 out of 17, leads to 136 combinations, 8 out of 17 leads to 24310 combinations), so instead we focused on a sampling strategy as proposed by Levine et al. [34]. It denotes the number of necessary trials n^* , given a confidence (in form of a z -score), a success probability of selecting a representative combination p and an error margin ϵ as follows:

$$n^* = \frac{\frac{(z\text{-score})^2 \cdot p \cdot (1-p)}{\epsilon^2} \cdot N}{\frac{(z\text{-score})^2 \cdot p \cdot (1-p)}{\epsilon^2} + N - 1}$$

With a minimal estimated success probability of 50% (0.5), an error margin of 5% (0.05) and a 95% confidence interval (translating to a 1.96 z -score), the overall

number of combinations to be investigated could be greatly reduced to just 14,427. So, considering the number of subjects and sessions per subject, the overall number of classifier trainings and adaptations was 288,540.

The absolute F_1 -scores are visualized in Figure 11.3, partitioned in three different scenarios: a *base* case, with only the starting sensor configuration (d dimensions) and the test-set being backprojected into the original input space (from $d+k$ dimensions to d dimensions), a comparison-case (*base+opp*), showing the potential of the paradigm, if labeled information in $d+k$ dimensions were available from the start, and our *method* of improvement, either CMM-based (Figure 11.3a) or DT-based (Figure 11.3b). From the graphs can be seen, that the performance of the base system can

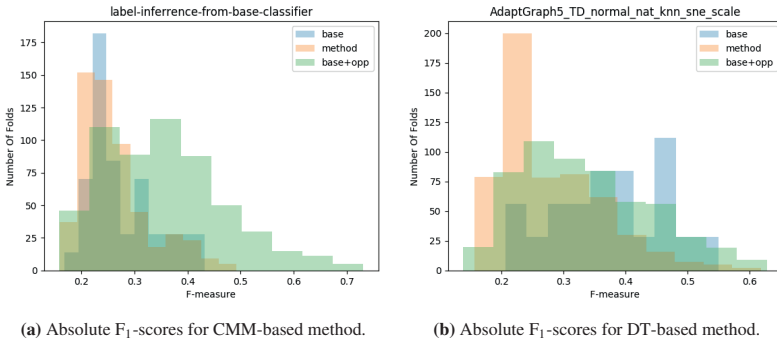
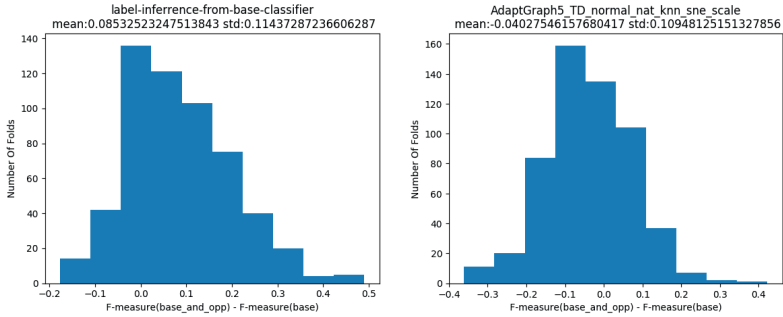


Fig. 11.3. F_1 -scores of base classifier, autonomously improved versions and a best-case comparison classifier.

be improved in some cases. Obviously, it is desirable to shift the overall distribution of classifier performances further to the right, so that it better matches to the “upper bound” (comparison, base+opp). In the case of a DT as basis, the improvements are not as frequent; which is a challenge we currently address by fine-tuning the parameters of the approach.

The relative improvements of our approaches can be seen in Figure 11.4. Again, the results on the left (Figure 11.4a) show the mean improvement and distribution of improvements over all experiments for the CMM-based approach, while the right chart visualizes the same for the DT-based approach (Figure 11.4b). From these graphs can be seen, how great the improvement can be expected to be. In the case of CMM, the expected improvements are 0.085 ± 0.113 on average, while in the case of a DT as basis, the improvements are currently deteriorations with an average of -0.040 ± 0.109 , due to bad parameter choices (the approach is feasible as we know from, e.g., [2]). It should also be mentioned, that the fairly high standard deviation for the CMM-based approach hints at a great deal of random influence, which is also expected to be reducible with further parameter investigations.



(a) Relative improvements with respect to the F_1 -score for the CMM-based method. (b) Relative improvements with respect to the F_1 -score for the DT-based method.

Fig. 11.4. Relative F_1 -score improvements.

11.5 Conclusion & Outlook

In this paper we proposed an overall approach of using generative modeling techniques to capture patterns in time series and enable different self-improving mechanisms for AR-classifiers. We compared the improvements of CMM- as well as DT-based methods with two comparison scenarios: One being a base case, answering the question *How would the performance of the system be without changes?* and the second comparison-case, answering the question *What is the maximum achievable performance, given all labeled samples from all input dimensions from the start?*

As shortly discussed before, both approaches can still be improved by parameter searches, as 1.) the CMM-based results show a rather high standard deviation (which hints for random influences) and 2.) the DT-based results showing performance degradations in some cases, while we see the applicability of DTs for improvement scenarios given, due to preliminary work ([2]).

For our future work we will increase the parametrization efforts for our current approaches and apply them to further benchmark datasets. Furthermore, we will apply recurrent neural networks with Long Short Term Memory (LSTM)-components to the problem, to gain more comparable results for these adaptation scenarios, but with a fresh approach.

Acknowledgements

The authors would like to thank the German research foundation (Deutsche Forschungsgemeinschaft, DFG) for the financial support in the context of the “Organic Computing Techniques for Runtime Self-Adaptation of Multi-Modal Activity Recognition Systems” project (SI 674/12-1, LU 1574/2-1).

References

1. Abowd, G., Dey, A., Brown, P., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Proc. of HUC. pp. 304–307. Springer (1999)
2. Bannach, D.: Tools and Methods to Support Opportunistic Human Activity Recognition. Ph.D. thesis, University of Kaiserslautern (2015)
3. Bannach, D., Jänicke, M., Rey, V.F., Tomforde, S., Sick, B., Lukowicz, P.: Self-adaptation of activity recognition systems to new sensors. CoRR abs/1701.08528 (2017), <http://arxiv.org/abs/1701.08528>
4. Banos, O., Damas, M., Pomares, H., Rojas, I.: On the use of sensor fusion to reduce the impact of rotational and additive noise in human activity recognition. *Sensors Journal* 12.6(6), 839–854 (2012)
5. Baum, L.E., Petrie, T.: Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics* 37(6), 1554 – 1563 (Dec 1966)
6. Becker, C., Schiele, G., Gubbels, H., Rothermel, K.: Base – a micro-broker-based middleware for pervasive computing. In: Proc. of PerCom. pp. 443–451 (2003)
7. Becker, J., Brändle, K., Brinkschulte, U., Henkel, J., Karl, W., Köster, T., Wenz, M., Wörn, H.: Digital on-demand computing organism for real-time systems. In: Proc. of ARCS. vol. 81, pp. 230–245 (2006)
8. Bishop, C.M.: *Pattern Recognition and Machine Learning*, chap. 1 Introduction, pp. 1 – 66. Springer, New York, NY (2006)
9. Brockmann, W., Buschermöhle, A., Hülsmann, J.: A generic concept to increase the robustness of embedded systems by trust management. In: Proc. of SMC. pp. 2037–2044 (2010)
10. Calatroni, A., Roggen, D., Tröster, G.: A methodology to use unknown new sensors for activity recognition by leveraging sporadic interactions with primitive sensors and behavioral assumptions. In: Proc. of UbiComp (2010)
11. Chavarriaga, R., Bayati, H., Millán, J.: Unsupervised adaptation for acceleration-based activity recognition: robustness to sensor displacement and rotation. *Personal and Ubiquitous Computing* 17(3), 479–490 (2013)
12. Chavarriaga, R., Sagha, H., Millán, J.D.: Ensemble creation and reconfiguration for activity recognition: An information theoretic approach. In: Proc. of SMC. pp. 2761–2766 (2011)
13. Chen, L., Hoey, J., Nugent, C.D., Cook, D.J., Yu, Z.: Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42(6), 790–808 (Nov 2012)
14. Clarkson, B., Mase, K., Pentland, A.: Recognizing user context via wearable sensors. In: Proc. of ISWC. pp. 69–75 (2000)
15. Durbin, R.: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, MA, USA (Apr 1998)
16. Faruque, M., Jahn, J., Ebi, T., Henkel, J.: Runtime thermal management using software agents for multi- and many-core architectures. *IEEE Design & Test of Computers* 27(6), 58–68 (2010)
17. Gellersen, H., Schmidt, A., Beigl, M.: Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications* 7(5), 341–351 (2002)
18. Henpraserttae, A., Thiemjarus, S., Marukatat, S.: Accurate activity recognition using a mobile phone regardless of device orientation and location. In: Proc. of BSN. pp. 41–46 (2011)

19. Hoffmann, M., Wittke, M., Bernard, Y., Soleymani, R., Hähner, J.: Dmctrac: Distributed multi camera tracking. In: Proc. of ICDSC. pp. 1–10 (2008)
20. Jakimovski, B., Maehle, E.: In situ self-reconfiguration of hexapod robot oscar using biologically inspired approaches. *Climbing and Walking Robots* pp. 11–332 (2010)
21. Jänicke, M.: Self-adapting multi-sensor system using classifiers based on gaussian mixture models. In: *Organic Computing: Doctoral Dissertation Colloquium 2015*. vol. 7, p. 109. kassel university press GmbH (2015)
22. Jelinek, F.: *Statistical Methods for Speech Recognition*. Language, Speech and Communications Series, MIT Press, Cambridge, MA, USA (Jan 1998)
23. Jiang, J.: A literature survey on domain adaptation of statistical classifiers. URL: <http://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey> (2008)
24. Jänicke, M., Sick, B., Tomforde, S.: Self-Adaptive Multi-Sensor Activity Recognition Systems Based on Gaussian Mixture Models. *Informatics* 5(3), 38 (Sep 2018), <http://www.mdpi.com/2227-9709/5/3/38>
25. Kunze, K., Lukowicz, P.: Dealing with sensor displacement in motion-based onbody activity recognition systems. In: Proc. UbiComp '08. pp. 20–29. Seoul, South Korea (2008)
26. Kunze, K., Lukowicz, P., Junker, H., Tröster, G.: Where am i: Recognizing on-body positions of wearable sensors. *LNCS 3479*, 264–275 (2005)
27. Kunze, K., Lukowicz, P., Partridge, K., Begole, B.: Which way am i facing: Inferring horizontal device orientation from an accelerometer signal. In: Proc. of ISWC. pp. 149–150 (2009)
28. Kurz, M., Ferscha, A.: Sensor abstractions for opportunistic activity and context recognition systems. In: *Smart Sensing and Context*, pp. 135–148. Springer (2010)
29. Kurz, M., Hölzl, G., Ferscha, A., Calatroni, A., Roggen, D., Tröster, G.: Real-time transfer and evaluation of activity recognition capabilities in an opportunistic system. In: Proc. of the 3rd Int. Conf. on Adaptive and Self-Adaptive Systems and Applications. pp. 73–78 (2011)
30. Kurz, M., Hölzl, G., Ferscha, A., Calatroni, A., Roggen, D., Tröster, G., Sagha, H., Chavarriaga, R., Millán, J.d.R., Bannach, D., et al.: The opportunity framework and data processing ecosystem for opportunistic activity and context recognition. *International Journal of Sensors, Wireless Communications and Control, Special Issue on Autonomic and Opportunistic Communications I* (2011)
31. Lane, N., Mohammod, M., Lin, M., Yang, X., Lu, H., Ali, S., Doryab, A., Berke, E., Choudhury, T., Campbell, A.: BeWell: A smartphone application to monitor, model and promote wellbeing. In: Proc. of PervasiveHealth (2011)
32. Lane, N., Xu, Y., Lu, H., Campbell, A., Choudhury, T., Eisenman, S.: Exploiting social networks for large-scale human behavior modeling. *IEEE Pervasive Computing* 10(4), 45–53 (2011)
33. Lester, J., Choudhury, T., Borriello, G.: A Practical Approach to Recognizing Physical Activities. *LNCS 3968*, 1 (2006)
34. Levine, D.M., Berenson, M.L., Stephan, D., Lysell, D.: *Statistics for managers using Microsoft Excel*, vol. 660. Prentice Hall Upper Saddle River, NJ (1999)
35. Lukowicz, P., Junker, H., Tröster, G.: Automatic calibration of body worn acceleration sensors. In: *Pervasive Computing*, pp. 176–181. Springer Berlin Heidelberg (2004)
36. Lukowicz, P., Pentland, S., Ferscha, A.: From context awareness to socially aware computing. *IEEE Pervasive Computing* 11(1), 32–41 (2012)
37. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the tota middleware. In: Proc. of PerCom. pp. 263–273 (2004)

38. Manning, C.D., Schütze, H.: *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA (May 1999)
39. Maurer, U., Rowe, A., Smailagic, A., Siewiorek, D.: eWatch: A Wearable Sensor and Notification Platform. In: *Proc. of BSN (2006)*
40. Mayora, O., Amrich, B., Bardram, J., Drager, C., Finke, A., Frost, M., Giordano, S., Gravenhorst, F., Grunerbl, A., Haring, C.: Personal health systems for bipolar disorder anecdotes, challenges and lessons learnt from MONARCA project. In: *Proc. of PervasiveHealth*. pp. 424–429 (2013)
41. Mostaghim, S., Schmeck, H., Wünsche, M., Geimer, M., Kautzmann, T.: Organic computing in off-highway machines. In: Müller-Schloer, C., Schmeck, H., Ungerer, T. (eds.) *Organic Computing – A Paradigm Shift for Complex Systems, Autonomic Systems*, vol. 1, pp. 601–603. Springer Basel (2011)
42. Müller-Schloer, C., Schmeck, H., Ungerer, T.: *Organic Computing – A Paradigm Shift for Complex Systems*. Birkhäuser (2011)
43. Müller-Schloer, C., Tomforde, S.: *Organic Computing – Technical Systems for Survival in the Real World*. Birkhäuser (2017), <https://doi.org/10.1007/978-3-319-68477-2>
44. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2), 257 – 286 (Feb 1989)
45. Rey, V.F., Lukowicz, P.: Label propagation: An unsupervised similarity based method for integrating new sensors in activity recognition systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1(3), 94:1 – 94:24 (Sep 2017)
46. Roggen, D., Forster, K., Calatroni, A., Holleczeck, T., Fang, Y., Troster, G., Lukowicz, P., Pirkl, G., Bannach, D., Kunze, K., et al.: Opportunity: Towards opportunistic activity and context recognition systems. In: *Proc. of WoWMoM*. pp. 1–6 (2009)
47. Roggen, D., Tröster, G., Lukowicz, P., Ferscha, A., Millán, J.D., Chavarriaga, R.: Opportunistic human activity and context recognition. *Computer* 46(2), 36–45 (2013)
48. Roggen, D., Calatroni, A., Rossi, M., Holleczeck, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Creatura, M., del R. Millán, J.: Collecting complex activity data sets in highly rich networked sensor environments. In: *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS)*, Kassel, Germany. IEEE Computer Society Press (June 2010)
49. Roth, M., Schmitt, J., Kiefhaber, R., Kluge, F., Ungerer, T.: Organic computing middleware for ubiquitous environments. In: *Organic Computing – A Paradigm Shift for Complex Systems*, pp. 339–351. Springer (2011)
50. Russomanno, D., Kothari, C., Thomas, O.: Building a sensor ontology: A practical approach leveraging iso and ogc models. In: *Proc. of IC-AI*. pp. 637–643 (2005)
51. Sheth, A., Henson, C., Sahoo, S.S.: Semantic sensor web. *IEEE Internet Computing* 12(4), 78–83 (2008)
52. Srinivasan, J., Adve, S., Bose, P., Rivers, J.: The impact of technology scaling on lifetime reliability. In: *DSN*. pp. 177–186 (2004)
53. Stiefmeier, T., Roggen, D., Ogris, G., Lukowicz, P., Tröster, G.: Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing* 7(2), 42–50 (2008)
54. Tapia, E., Choudhury, T., Philipose, M.: Building reliable activity models using hierarchical shrinkage and mined ontology. In: *Pervasive Computing*. pp. 17–32. Springer (2006)
55. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic computing in the spotlight. *CoRR* abs/1701.08125 (2017), <http://arxiv.org/abs/1701.08125>
56. Tsuda, K., Kawanabe, M., Rätsch, G., Sonnenburg, S., Müller, K.R.: A new discriminative kernel from probabilistic models. *Neural Computation* 14(10), 2397 – 2414 (Oct 2002)

ISBN 978-3-7376-0945-6



9 783737 609456 >