Self-aware Optimization of Adaptation Planning Strategies

VERONIKA LESCH and MARIUS HADRY, University of Würzburg, Germany
CHRISTIAN KRUPITZER, University of Hohenheim, Germany
SAMUEL KOUNEV, University of Würzburg, Germany

In today's world, circumstances, processes, and requirements for software systems are becoming increasingly complex. To operate properly in such dynamic environments, software systems must adapt to these changes, which has led to the research area of Self-Adaptive Systems (SAS). Platooning is one example of adaptive systems in Intelligent Transportation Systems, which is the ability of vehicles to travel with close inter-vehicle distances. This technology leads to an increase in road throughput and safety, which directly addresses the increased infrastructure needs due to increased traffic on the roads. However, the No-Free-Lunch theorem states that the performance of one adaptation planning strategy is not necessarily transferable to other problems. Moreover, especially in the field of SAS, the selection of the most appropriate strategy depends on the current situation of the system. In this article, we address the problem of self-aware optimization of adaptation planning strategies by designing a framework that includes situation detection, strategy selection, and parameter optimization of the selected strategies. We apply our approach on the case study platooning coordination and evaluate the performance of the proposed framework.

CCS Concepts: • **Software and its engineering** → **Layered systems**; **Development frameworks and environments**; • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies** → **Simulation evaluation**; • **Theory of computation** → *Design and analysis of algorithms;*

Additional Key Words and Phrases: Self-awareness, optimization, cyber-physical systems, adaptation planning strategies, platooning, framework

ACM Reference format:
Veronika Lesch, Marius Hadry, Christian Krupitzer, and Samuel Kounev. 2022. Self-aware Optimization of Adaptation Planning Strategies. *ACM Trans. Autonom. Adapt. Syst.* 00, JA, Article 00 (November 2022), 35 pages.
https://doi.org/10.1145/3568680

1 INTRODUCTION

In a world as dynamic as we find it today, where circumstances, processes, and requirements are becoming increasingly complex, the challenges for software systems to be able to work in these dynamic environments are also increasing. One of the most critical challenges for these systems is to analyze their environment and to adapt to changes accordingly. The **Self-Adaptive System (SAS)** [11, 33] research area addresses these challenges. The SAS can change their behavior

Authors' addresses: V. Lesch (corresponding author), M. Hadry, and S. Kounev, University of Würzburg, Würzburg, Germany; emails: {veronika.lesch, marius.hadry, samuel.kounev}@uni-wuerzburg.de; C. Krupitzer, University of Hohenheim, Hohenheim, Germany; email: christian.krupitzer@uni-hohenheim.de. Q1

1556-4665/2022/11-ART00 $15.00
https://doi.org/10.1145/3568680

and deal with changes in their environment and the system itself [35]. In our daily lives, we are constantly in contact with SAS that aim to support and improve our way of life without us directly noticing it. One SAS use case from **Intelligent Transportation Systems (ITS)** are electric traffic signals that have led to the development of real-time traffic control in urban areas [66]. Another promising example for ITS is platooning, which addresses increased infrastructure needs resulting from increased traffic on roads. Due to advances in autonomous driving, an increased infrastructure need can be reduced through platooning, which is the ability of vehicles to travel with very close inter-vehicle distances, enabled by communication [52]. The use of platooning increases road throughput [4] and safety [52]. Platooning coordination is the process of assigning vehicles to platoons and controlling the platooning activities. The platooning coordination problem is a multi-objective problem with several dimensions, since objectives of the drivers, aspects of the platoon, and global traffic need to be considered [57]. Platoons are usually coordinated using platooning coordination strategies. This coordination is an example of SAS in ITS, as these coordination strategies can be considered as adaptation planning strategies that adapt the system, in this case the platoons, to their current state and environment.

In line with the No-Free-Lunch theorem [65] the proper selection of adaptation planning strategies is a key factor in the success of any SAS, as the performance of one strategy may not necessarily be transferable to other application scenarios. In the year 1976, John R. Rice already defined the algorithm selection problem, which involves finding the best-performing algorithm for the current problem [51]. This leads to the idea of a mechanism that automatically selects the most promising algorithm that is also generalizable to be applied in a variety of applications. The observation of a situation-dependent adaptation planning strategy in self-adaptive systems [11, 17, 33], which was experimentally confirmed in our recent ACSOS publication [40], opens a wide area to which such a mechanism can be applied. Gathered observations can be used to apply different strategies in different situations or to adjust the parameters of a strategy. Furthermore, the knowledge can be used in combination with previous experiences to learn in which situation which strategy and which parameter configuration works best. This idea of combined reasoning and learning can be found in the **Self-aware Computing (SeAC)** research area, whose ideas and approaches will be applied in this work. There are several approaches to situation detection [8, 15, 22, 25, 43, 49, 53], algorithm selection [6, 26, 27, 29, 55], and parameter optimization [12, 16, 46, 62, 67] especially in the SAS literature. However, there is no integrated approach that combines these ideas into a mechanism that is generalizable and applicable to a variety of use cases.

As the results of our ACSOS publication [40] confirm the situation-dependent performance of adaptation planning strategies, we propose a self-aware framework for selecting and optimizing adaptation planning strategies in this article. The framework explicitly addresses situation-dependent behavior of these strategies by automatically identifying the current situation, selecting the most promising strategy, and optimizing the parameter of the selected strategies. In addition, the framework applies concepts from SeAC research and is able to learn and reason from previous decisions and experiences. Our framework is intended for application in diverse use cases for which a specific adapter component enables generic applicability. To showcase the functionality and analyze the performance of the framework, we apply it on the platooning coordination use case. Therefore, we define three platooning coordination strategies and apply Bayesian optimization for parameter tuning. As evaluation environment, we use the platooning simulation framework presented in Reference [32] that integrates the platooning simulator Plexe [54], which is based on Veins [56] (including SUMO and Omnet++) with the tool **Platooning Coordination System (PCS)** [34].

The remainder of this article is organized as follows: Section 2 discusses related work. Section 3 presents our running example platooning coordination and summarizes our previous results.

Section 4 proposes our self-aware framework before the subsequent sections present the details of the Coordination (cf. Section 5), the Domain Data Model (cf. Section 6), the Situation Detection (cf. Section 7), the Strategy Selection (cf. Section 8), and the Parameter Optimization (cf. Section 9) components. Section 10 presents the evaluation of the framework on the platooning coordination use case. Finally, Section 11 summarizes the article and outlines future work.

2 RELATED WORK

Several works exist that address situation-awareness, meta self-awareness, algorithm selection, and meta optimization. In the following, we summarize most important findings in these areas and discuss their relatedness to this work. A recent study by Calinescu et al. [8] has shown that situation-awareness is the main driver for the development of self-adaptive systems and is therefore still an important research topic with many open research challenges. Endsley [15] presents a theoretical model of situation-awareness in relation to dynamic human decision-making, building on research on naturalistic decision-making. Fredericks et al. [17] present an approach that uses clustering to determine the current situation. They use this information for optimization techniques to discover the optimal configuration for black-box systems. Liu et al. [43] propose an approach to situation-awareness in autonomous driving that aims to improve the decision-making process in an urban environment. Rockl et al. [53] propose an architecture for driver assistance systems that uses increased environmental information to detect hazardous situations. Hardes et al. [23] address communication problems in urban platooning scenarios by using the concept of situation-awareness. Porter et al. [49] propose a software framework that learns optimal system assemblies in emergent software systems. Kang et al. [25] analyze which history length and sensor range provide the best results for long-term situational awareness. Finally, we analyze in our previous study the situation-awareness of adaptation planning strategies in the platooning use case [40]. In this article, we use the mentioned publications as inspiration to create a situation-awareness component for our framework (see Section 7). Especially, the work of Fredericks et al. [17], which also uses clustering techniques to identify situations and our previous paper [40], which is the foundation for our rule-based situation detection are highly related to our approach.

According to Lewis et al. [41], meta-self-awareness "leads to the ability to model and reason about changing tradeoffs during the system's lifetime." Cox et al. [13] research on meta-cognition, which bridges psychology and computer science. Agarwal et al. [3] provide an approach that allows computer systems to reason about their own knowledge. Perrouin et al. [48] propose a rule-based approach to meta-self-awareness. They use layered MAPE-K control loops to optimize adaptation decisions and make an adaptive system "resilient to a larger number of unexpected situations" [48]. Gerostathopoulos et al. [18] propose the concept of meta-adaption for cyber-physical systems, which improves the adaptation of a cyber-physical system by generating new self-adaptation strategies at runtime. Kinneer et al. [28] propose the idea of re-using knowledge from previous plans for optimization. They use a white-box approach with knowledge about the system combined with a genetic algorithm to respond to unexpected adaptation scenarios. Similar to the previous paragraph, we also use existing literature in meta-self-awareness as inspiration for our framework. Especially the definition from Lewis et al. [41] and the idea of layered MAPE-K loops from Perrouin et al. [48] led us to our concept of a generic optimization framework, as presented in Section 4.

Kate Smith-Miles considers algorithm selection as learning problem [55]. She reviews the interdisciplinary literature dealing with algorithm selection and presents the developments in this research area. Kerschke et al. provide a survey on automated algorithm selection [26]. The survey covers early and recent work in this area and discusses promising application areas. Further, it includes an overview on related areas such as algorithm configuration and scheduling. Pascal Kerschke and Heike Trautmann contribute an approach for automatic model construction for algorithm selection in continuous black-box optimization problems [27]. The goal of this approach is to reduce the required resources of the selected optimization algorithms. Kotthoff et al. apply algorithm selection on the TSP problem [29]. They apply two existing TSP solvers and show that they perform complementary in different instances. The authors design algorithm selectors based on existing TSP features from the literature as well as new features. Bischl et al. propose a benchmark library for algorithm selection [6]. They define a standardized format for representing algorithm selection scenarios. Further, they provide a repository containing data sets from the literature to compare proposed approaches. The literature on algorithm selection already provides definitions, surveys, and a large set of approaches to address the algorithm selection problem. We used this literature in our research to generate an idea how the information of the current situation can be used to select a promising adaptation planning strategy and to learn from earlier decisions. However, we did not use any of the proposed methods directly in our component, as described in Section 8.

Neumüller et al. [46] present an implementation of parameter meta-optimization for the heuristic optimization environment *HeuristicLab Hive*. Their approach minimizes the expert knowledge required to adapt the parameters of a meta-heuristic. In their evaluation, Neumüller et al. showed that the obtained parameter combinations in some cases deviate strongly from the usual settings. However, their approach mainly covers single-objective optimization, whereas a multi-objective problem can only be assessed using a normalized and weighted sum of objectives. Feurer et al. [16] improve the Sequential Model-based Bayesian Optimization used for tuning the parameters of machine learning algorithms involving meta-learning. Using the knowledge from past optimization runs, they showed significant improvement in the Sequential Model-based Bayesian Optimization algorithm. Zhang et al. [67] address the problem of release planning, which means the process of deciding which features to integrate into the next version of a software release. The authors perform a study on various meta- and hyper-heuristics used for multi-objective release planning. They use different hyper-heuristic algorithms to decide on search operators for meta-heuristics to improve solution quality and compare their performance. Chis et al. [12] use the Framework for Automatic Design Space Exploration to compare the performance of different multi-objective meta-heuristics. The authors show that all algorithms find similar Pareto front approximations with good solution quality. Similarly, Vinctan et al. [62] deal with design space exploration by implementing a meta-optimization layer for the tool Framework for Automatic Design Space Exploration. With this approach, it is possible to introduce a meta-optimization function that can use multiple meta-heuristics simultaneously by switching between them at simulation runtime. In the evaluation, the authors show that their meta-optimization approach leads to better results than running two different meta-heuristics independently and combining their results. The presented literature of this paragraph covers the terms meta-optimization and parameter tuning. We used the existing literature to search for a promising approach for parameter tuning. According to the literature, we decided to integrate Bayesian optimization into our Parameter Optimization Component Section 9 as a promising starting point for our prototype.

Another research direction related to this work is the area of Auto-ML. As the name suggests, automated machine learning focuses on automating machine learning mechanisms by using pipelines in combination with hyperparameter optimization to reduce manual effort. Reinbo, for example, is an Auto-ML framework that uses task pipelines and implements reinforcement learning and Bayesian optimization to automatically determine the parameters [59]. A similar approach is used by Chai et al., who propose an Auto-ML framework that covers the common problem of data drift in machine learning [9]. Thornton et al. propose a mechanism for hyper-parameters selection and optimization in the context of classification algorithms [60]. Finally, Li et al. attempt to solve the problem of tuning hyper-parameters using a random search mechanism combined with adaptive resource allocation and early-stopping [42]. Similar to the previous paragraph, the literature on Auto-ML also tries to optimize hyperparameter automatically. This literature also showed us that Bayesian optimization is a promising technique when it comes to reducing manual effort for parametrization. This insight further strengthened our decision to use Bayesian optimization in Section 9.

This work delineates from the presented related work as follows: All mentioned approaches already cover aspects of our proposed framework, such as a rule-based meta-self-aware approach, situation-awareness, determining the optimal configuration of a system, or performance comparison of optimization techniques. However, there is no other work that integrates all these aspects into one framework to simplify and fasten development and application of self-adaptivity of systems in combination with a separation of concerns. The combination of a multi-layered framework with the LRA-M control loop and the integration of adaptation planning strategies, situation-awareness, strategy selection, learning approaches, and optimization techniques make the proposed approach unique and a valuable contribution to the research community.

3 RUNNING EXAMPLE: PLATOONING COORDINATION

In this section, we introduce our running example platooning coordination as meaningful example of adaptation planning systems. Then, we summarize findings of our previous publication [40] and discuss the contributions of this article.

Platooning is the ability of vehicles to travel with very close inter-vehicle distances, enabled by communication [52]. The use of platooning can reduce fuel consumption through slipstream effects, increases road throughput [4] through homogenization of traffic, and can reduce the likelihood of traffic congestion and accidents and, thus, increases safety [52]. In our use case, we distinguish two levels of platooning [36]:

(1) **Platooning control** captures the control of a single vehicle on the lowest possible level (e.g., distance maintenance, braking, overtaking).
(2) **Platooning coordination** includes the management of (i) the composition of a platoon, (ii) inter-platoon interactions, as well as (iii) interactions between other vehicles and platoons.

While the feasibility of platooning control is shown in diverse projects, the issue of platooning coordination under real conditions and constraints still exists [36]. The platooning coordination problem is a multi-objective problem with diverse dimensions, since objectives of the drivers, aspects of the platoon, and global traffic need to be considered [57] as well as fairness between participants must be guaranteed, as the leading vehicle benefits less from slipstream effects [38]. To address this problem, platooning coordination strategies aim at adapting the overall traffic system with regards to the mentioned goals and objectives.

Following the observation from Reference [17] that the choice of the algorithm for adaptation planning in self-adaptive systems [11, 33] depends on the situation of the system, we claimed in our previous paper that the choice of the platooning coordination strategy also is situation-dependent [40]. In the mentioned paper, we analyzed different platooning coordination strategies and optimization algorithms for parameter tuning under varying traffic situations to show the usefulness of combining a situation-dependent choice of the adaptation planning strategy with an optimization of the parameters. Following this idea, our previous paper provided three contributions: (i) definition of a three-layered system model for self-aware optimization in self-adaptive systems, (ii) analysis of a set of platooning coordination strategies to identify situation-dependent performance and strategy-dependent optimization techniques, and (iii) a reusable testbed for evaluating meta-optimized adaptation planning strategies.

The extensive case study of our previous paper [40] revealed three important findings regarding the selection of platooning coordination strategies, their parameterization, and the performance of optimization techniques in this context. First, we identified that the choice of strategy depends on the addressed objectives and none of the strategies performed best for all metrics. Second, we confirmed our claim that the performance of platooning coordination strategies depends on the current situation and its parametrization. Third, our analysis showed that Bayesian parameter optimization improves the performance best and fastest compared to other optimization approaches. In summary, we concluded that the choice of the adaptation planning strategy but also the strategy's parameters is not a "one fitting all" choice, especially in multi-objective scenarios.

This article bases on our previous findings and extents the proposed contributions significantly. We now propose a self-aware optimization framework for adaptation planning strategies that is not limited to the platooning use case but can be applied on a wide variety of self-adaptive use cases. This framework is able to analyze the current situation, select the most promising adaptation strategy, and perform its parameters. Further, it integrates self-aware concepts and learns and reasons from previous decisions and experiences.

4 SELF-AWARE OPTIMIZATION OF ADAPTATION PLANNING STRATEGIES

This section proposes the framework for self-aware optimization of adaptation planning strategies. Section 4.1 summarizes assumptions, and Section 4.2 presents the system model. Afterwards, Section 4.3 provides an overview of the framework composition, and Section 4.4 describes the use-case-specific adapter for linking the framework to any **cyber-physical system (CPS)** use case. Finally, Section 4.5 discusses the application of self-awareness concepts.

4.1 Assumptions

In this section, we state assumptions for the design of the framework to ensure broad applicability in various use cases. The following assumptions ensure the proper operation of the framework as well as the use case and define interactions between both systems. At the same time, they point out limitations that can be addressed in future work.

First, we assume that use cases consist of an environment with operating entities and an adaptation planning system. The entities operate based on their individual goals and actions, report observations regularly, and adhere to a given plan from the adaptation planning system. The adaptation planning system monitors entities and plans adaptation actions based on global goals, where applied strategies and parameters can be changed at runtime. This structure and obedience of the entities for a centralized decision-making management which can rely on the executing adaptation planning system. Second, we assume a digitized use case that captures performance and monitoring data about itself and is able to transmit it to a defined management entity performing higher-level optimizations. At the moment, we assume a flawless communication and interaction between use case and management entity that makes a control mechanism for communication unnecessary. This assumption severely limits the direct applicability of the framework at the moment. However, we are convinced that a reasonable choice of communication and transmission technologies can put this limitation into perspective. In the worst-case consideration with respect to no perfect communication, the framework no longer receives observations from the use case and can no longer make adaptation decisions. Additionally, the decisions may no longer be transmitted to the use case. However, this in no way restricts the general operation of the use case, as it executes a working adaptation strategy at all times even without adaptation decisions from the framework. Third, we assume that the adaptation planning system works independently of a

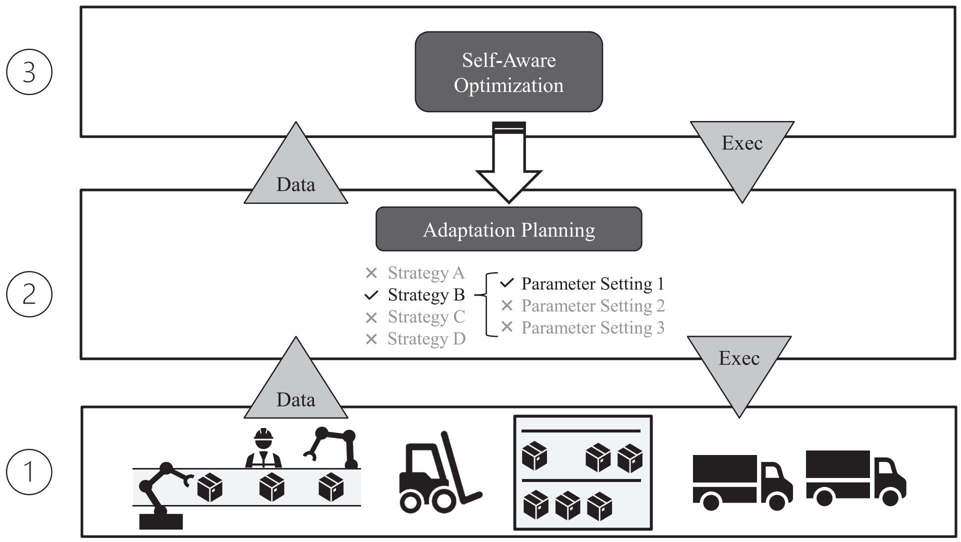


Fig. 1. Multi-layer architecture of the self-aware optimization framework. Layer 1 represents an adaptive system, the adaptation planning system is shown in Layer 2, and Layer 3 shows the self-aware optimization.

higher-level optimization, i.e., the framework, and can be used with a previously defined strategy algorithm and parameter set to remain functional even when management is not available. Finally, we assume that the framework provides optimized decisions to the adaptation planning system that retrieves and successfully implements these changes. This excludes the control of instructed changes by the management and allows us to fully focus the development on the functionalities of the framework.

4.2 System Model

This section introduces the generalized system model applicable on a variety of CPS use cases to define the self-aware optimization framework. Our system model follows the three-layer approach from Kramer and Magee [31] to incorporate the principles of maintainability and separation of concerns. Further, it applies the Hierarchical Control Pattern from Weyns et al. in which "different levels of abstraction [. . .] may operate at different time scales" [64, p.93]. Figure 1 presents the three layers (i) application, (ii) adaptation planning, and (iii) self-aware optimization, which we explain in the following: We refer to the bottom layer ① of the system model as the **application layer** and consider real-world CPS use cases as the managed system. Entities of the use case monitor themselves and their environment and report observations to the next layer. After an adaptation planning cycle, the use case entities can receive adaptation actions to follow and execute.

The middle layer ②, called **adaptation planning**, includes the adaptation planning system. It receives observations from the application and applies a strategy with given parameter settings to determine adaptation actions. We name the adaptation planning strategies this way to clearly delineate them from other applied algorithms used in the framework, which is the third layer. In fact, technically spoken, these adaptation planning strategies are algorithms that receive data from the use case, analyze the proper operation of the use case, and plan adaptation decisions that will be given to the use case. In terms of the platooning coordination use case, the entities in the use case are the vehicles, and the platooning coordination algorithms can be considered as adaptation planning strategies. We stick to this abstract naming of adaptation planning strategies to delineate

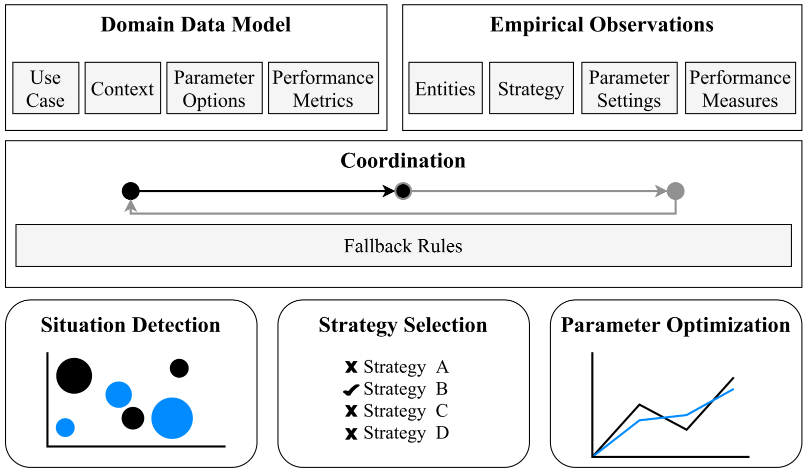


Fig. 2. Composition of the self-aware optimization framework. The framework contains the *DDM* for configuration, the Empirical Observations as a repository, a *Coordination* component that manages the workflow, and the three main components *Situation Detection*, *Strategy Selection*, and *Parameter Optimization*.

from running algorithms in the framework and further remain independent from use case details. We assume that the user of the framework provides multiple strategies, customized for the particular use case, to provide the possibility of strategy exchange when needed. The performance data of the selected strategy and application monitoring data is transferred to the next layer. After a self-aware optimization cycle, the adaptation planning layer may receive instructions to change the strategy parametrization or to replace the strategy.

Finally, the third layer ③ is called **self-aware optimization** and is responsible for optimizing strategy parameters and selecting the best-fitting strategy for the adaptation planning system. It incorporates three components: (i) *Situation Detection*, (ii) *Strategy Selection*, and (iii) *Parameter Optimization*. The *Situation Detection* component receives monitoring data, that is, the application observations and performance data from layer ② and categorizes the observations into a currently present situation. The *Strategy Selection* component uses this categorization, combines it with experience from similar situations in the past, and selects the most appropriate adaptation planning strategy. Finally, the *Parameter Optimization* component tunes the parameters of the adaptation planning strategy. A knowledge base manages the set of known situations as well as corresponding decisions and continuously learns which parameter and algorithm combination fits best for the situations already experienced.

4.3 Framework Composition

This section presents the composition of the generically applicable self-aware optimization framework, which is the third layer of our presented system model. The framework consists of several interacting components, as depicted in Figure 2. In the following, we briefly introduce each component and state its main contribution to the framework and outsource detailed descriptions of the components to the following sections.

Domain-Data-Model: The user of the framework can use the ***Domain-Data-Model*** **(DDM)** to configure the entire framework and all its components. It is the only part of the framework that the user needs to configure with use-case-specific information, and the framework considers the two lower levels as black box. The *DDM* contains information about the use case, context, parameter options, and performance metrics.

Empirical Observations: The second component of the framework is responsible for managing all sensor data received from the use case and is called *Empirical Observations*. It processes incoming data and provides an interface for the other components to retrieve relevant data for their current task.

Coordination: The central component of the framework is the *Coordination*, which is responsible for the regular operation of the framework. This component is constantly active, regularly invokes the other components of the framework, and delivers the required observation data. In the event that one of the other components fails, this component can fall back to user-defined rules to remain functional. Hence, this component's main responsibility is the coordination of all components so they work together in the intended way. This responsibility also includes tasks to synchronize the components, their required data, and the decisions made by the framework.

We agree with the reviewer that our Coordination component handles synchronization tasks between the different components and the received monitoring data. Its main responsibility is to make all components work together. Without the Coordination component, the whole framework would not be functional and hence, it has a crucial responsibility regarding the coordination of all components

Situation Detection: The *Situation Detection* component receives the observation data of the use case, such as the entities and their current state, and determines the current situation. So far, we apply clustering algorithms but the component can be extended with other approaches if required. After determining the situation, the component returns the situation ID.

Strategy Selection: The *Coordination* invokes the *Strategy Selection* component using the information of the current situation. This component combines knowledge about the current situation with experience from previous decisions in similar situations and determines the most appropriate adaptation planning strategy for the current situation. It returns the decision to the *Coordination* component.

Parameter Optimization The *Parameter Optimization* component receives the current parameter settings as starting point, historical data of the current situation, the corresponding adaptation planning algorithm, and performance measures. It performs an optimization process to tune the parameter setting for this adaptation planning strategy to the current situation. Afterwards, it returns the settings to the *Coordination* component.

In addition to the general composition of the framework, we illustrate the workflow of the framework as a sequence diagram in Figure 3. The user on the left side configures and starts the framework using the *DDM*, sets up the use case, and configures it. The use case starts its operation and sends the defined observations to the framework in regular intervals, regardless of the current computational state of the framework. The *Coordination* component of the framework processes incoming observations and forwards them to the *Empirical Observations*. After a certain number of received observations, the *Controller* component triggers the first execution of the *Situation Detection* component and forwards relevant observation data to this component. In the meantime, the *Coordination* component receives further observations from the use case, which are stored but not used until the next round of execution. The *Situation Detection* returns the situation ID to the *Coordination*, which updates the system model of the environment. Then, the *Coordination* component triggers the *Strategy Selection* with filtered observation data containing only observations of the identified situation. This component applies model-based reasoning, determines the most promising adaptation planning strategy, and returns it to the *Coordination* component, which updates the system model. Finally, the observed data is filtered again to include only data for the current situation and active strategy and triggers the *Parameter Optimization*. After the *Coordination* component obtains this parameter setting, it updates the system model and sends adaptation tasks to the adaptation planning system, which executes them. This step completes

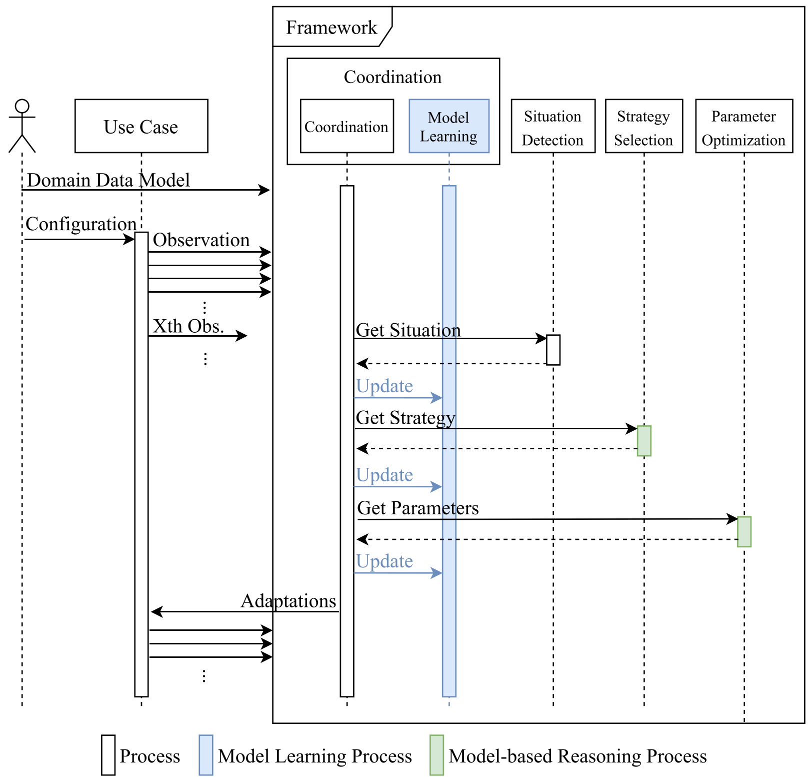


Fig. 3. Sequence diagram of the workflow of the self-aware optimization framework. The user configures the framework and the use case sends observations. The framework processes the observations, identifies the current situation, selects the strategy and parameter setting, and continuously learns and updates its models.

one round of execution in the framework and after a predefined waiting time, the *Coordination* starts the next round.

4.4 Use-case-specific Adapter of the Framework

All the components of the framework are designed to be generically applicable to a variety of use cases enabled by the *DDM* definition of use-case-specific characteristics and an adapter that manages the connection between use case and framework as described in the following. This section briefly summarizes the required user actions to apply the framework for any use case.

Figure 4 provides an overview of the architecture of the adapter required to connect the framework to any use case. The self-aware optimization framework is depicted at the top providing two REST APIs for receiving observations (on the left) and providing adaptation actions (on the right) that are defined using the *DDM*. The use case consisting of the two lower levels (see Section 4.2) is depicted at the bottom of the figure. The center of the figure presents two adapter components required to connect the components of the framework with use-case-specific system elements: (i) Data Preprocessing and (ii) Adaptation Executor. The Data Preprocessing component receives raw monitoring data from the use case, preprocesses this data, and potentially calculates additional aggregate metrics that may be required to assess the performance of the use case. The Adaptation Executor component, depicted on the center right of the figure, retrieves the adaptation decisions

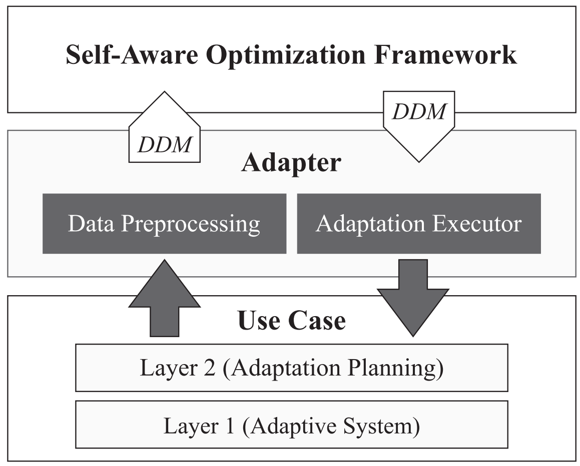


Fig. 4. Use case adapter for the generic self-aware optimization framework. The use case with its two layers adaptive system and adaptation planning are depicted at the bottom. It communicates with the Framework by sending observations and retrieving adaptation actions. Additional Data Preprocessing and Adaptation Executor components can provide a further abstraction level.

from the framework and converts them into specific adaptation actions for the use case. Since both adapter components handle data transfer to and from the framework based on REST APIs, the implementation effort required to apply them to a new use case is reduced. If the use case already provides the possibilities to send monitoring data directly to the framework and retrieve and execute adaptation decisions, then these adapter components may not be necessary.

In terms of communication load, the framework is designed to be able to reduce the overhead to an absolute minimum. This includes the transmission of already aggregated performance metrics from the use case to the framework and the adaptation information towards the use case. This can be achieved by observing the use case within the second layer and preprocessing and aggregating the performance metrics to the used form for the framework. Additionally, these aggregated metrics can be send batch-wise limited by the frequency the situation detection uses to identify changing situations. All these mechanisms can help to reduce the communication load between framework and use case.

4.5 Integrating Self-aware Computing

In this section, we present our concept of a self-aware optimization framework using a control loop to discuss the integration of SeAC. In line with the used self-awareness terminology, we focus this section on the corresponding LRA-M control loop [30]. Since this loop is a general-purpose concept applicable to diverse systems, we modify it to explicitly include the functionalities of our proposed framework, as shown in Figure 5.

The loop displays the system, also called the self, and its interfaces with the environment. It interacts with the environment by (i) perceiving *Phenomena* and storing them as *Empirical Observations*, (ii) receiving *Goals* to be achieved, and (iii) executing *Actions* based on the decisions made. The Empirical Observations are captured in the use case, i.e., the application layer of the system model, and used in the *Learn* and *Reason* modules. During the ongoing learning process, the observations are abstracted into models that contain knowledge about the two lower levels and recognize new situations. We add the **Situation Detection** component into the Learn module, which receives performance data of the managed use case with periodic observations and learns

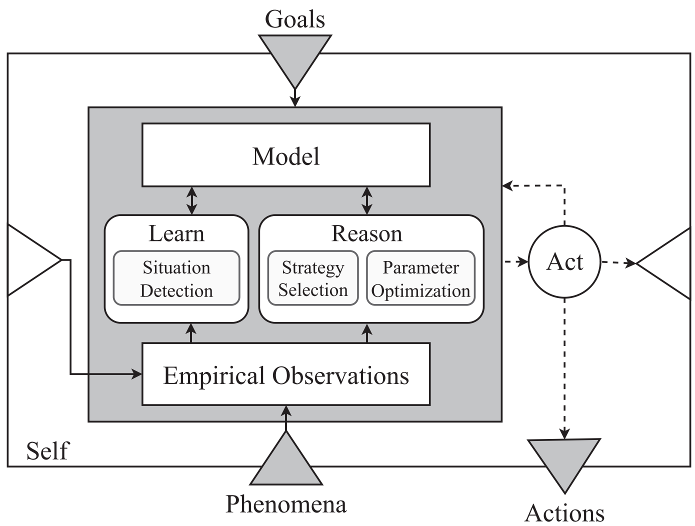


Fig. 5. Modified LRA-M control loop based on Kounev et al. [2017]. The basic LRA-M control loop is extended to include analysis and the meta-optimization in the Learn module and planning through optimization in the Reason module.

the impacts of the actions taken based on the current situation. Reasoning gives the framework the ability to consider which adaptation actions might be beneficial as reaction to changes in the environment or deteriorated performance values. Hence, we assign the two components (i) *Strategy Selection*, and (ii) *Parameter Optimization* to this module. The **Strategy Selection** component combines the information from *Situation Detection* and the current use case performance with the learned models about the use case and determines whether to keep the current strategy or switch to another existing strategy. The **Parameter Optimization** component applies optimization techniques using all observations from the current situation to tune the parameters for the selected strategy. 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. We present the details of all components in Sections 5 to 9.

5 COORDINATION COMPONENT

This section provides a more technical view of the **Coordination** component introduced in Section 4.3 and depicted in Figure 2. The pseudocode in Algorithm 1 summarizes the workflow of the *Coordination* component. The *Coordination* is responsible for initializing and invoking all other components of the framework. It processes incoming observations and updates the system models based on observations and the framework's adaptation decisions. It is triggered at the start of the framework and instantiates all components of the framework (Lines 1–2) according to the *DDM*. Whenever the required number of new observations are received, the *Coordination* component triggers a new round of execution. As a first step, the component uses received data to derive additional information relevant to subsequent processing (Line 3). We use the Hypervolume [63] to reduce the observed performance indicators of the use case to a single performance value. This allows us to use any single-objective optimization technique in the *Parameter Optimization* component without requiring multi-objectiveness for this technique. Afterwards, the component stores the observation and newly derived information in the Empirical Observations component (Line 4).

ALGORITHM 1: Pseudocode workflow of the Coordination component.

```
   Input: DDM, new observation, existing observations
1  if start of framework then
2      initialize components defined in the DDM;
3  derive additional information from the observation;
4  save new observation;
5  situation ← invoke Situation Detection on all observations;
6  if situation could not be determined then
7      adaptations ← apply fallback rules to all observations;
8      update system model with current adaptation decision;
9      send adaptations;
10 else
11     update system model with current situation;
12     if waiting time after previous adaptation action is over then
13         if same situation as before AND number of optimization attempts not met then
14             parameter ← invoke Parameter Optimization on observations of current situation and
               strategy;
15         else
16             strategy ← invoke Strategy Selection on observations of current situation;
17             parameter ← invoke Parameter Optimization on observations of current situation and
               strategy;
18         update system model with current adaptation decision;
19         send adaptation decision to use case;
```

Then, the *Coordination* passes the new observation to the *Situation Detection* component (Line 5), which applies clustering algorithms to identify the current situation. After the *Situation Detection* identified the current situation, it returns the situation to the *Coordination*. If the available observation data is not sufficient for the clustering algorithm or the current situation is clustered as noise, then the *Situation Detection* does not return a situation.

The *Coordination* component then checks whether the *Situation Detection* was successful (Line 6). If the *Situation Detection* did not return a situation, then the *Coordination* component applies the fallback rules to the current observations (Line 7). Then, the *Coordination* updates the model with the most recent adaptation decision (omitting this step if fallback rules are applied) and sends the adaptations to the use case (Lines 8–9). In case the *Situation Detection* returned a valid situation (Line 10), the *Coordination* updates information about the current situation to the model (Line 11). Afterwards, the *Coordination* checks whether the waiting time after a previous adaptation action has expired (Line 12). This user-defined waiting time serves as cool-down period for use case adaptations to take effect. If the waiting time is still active, then the current round of execution ends and the *Coordination* waits for the next observations. If the waiting time has expired, then new adaptation decisions can be sent to the use case. Therefore, the *Coordination* analyzes whether the currently active situation is similar to the previous one and whether the number of optimization attempts is not met (Line 13). If this holds, then the *Coordination* requests all observations of the current situation and strategy combination and passes them to the *Parameter Optimization*. The *Parameter Optimization* computes a new set of parameters and returns it (Line 14). However, if the number of optimization attempts has been exceeded, then this indicates poor performance of the currently used strategy, which results in a search for a new,

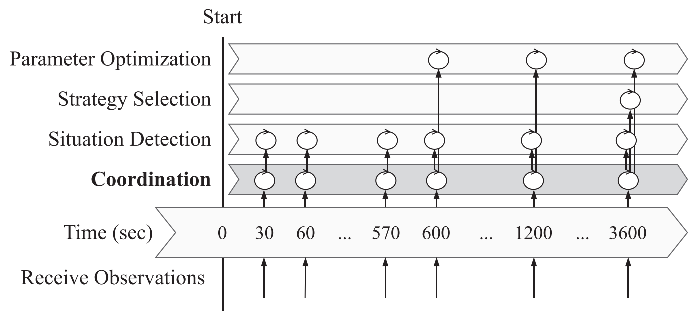


Fig. 6. Timescale of the components and their computations the *Coordination* invokes. Illustrated is a use-case-specific time scale of 3,600 seconds where observations arrive every 30 seconds. Each observation triggers an execution of the *Coordination*, which then decides which other components to invoke.

better-fitting strategy. In this case, or whenever the situation changed (Line 15), the *Coordination* requests all observations of the current situation and passes them to the *Strategy Selection* component (Line 16). This component uses this information to reason about the most promising strategy for adaptation planning and returns the selected strategy. Then, the *Coordination* requests all observations of the current situation and the selected strategy to pass them to the *Parameter Optimization* (Line 17). This component performs an optimization to select the most promising parameter settings for this strategy and returns the results. The *Coordination*, in turn, uses the strategy decision and its parameterization to update the model of the system (Line 18). Finally, it sends the adaptation decisions including the strategy and the parameter setting to the use case (Line 19).

To better understand the timing within the framework, we present an example timescale for invoking the three components *Situation Detection*, *Strategy Selection*, and *Parameter Optimization* in Figure 6. All timing values can be defined by the user with respect to the use case. Therefore, the timing presented here should only be considered as an example for demonstration and not as the fixed timing of the framework for all use cases. For simplicity, we assume that no situation changes occur in this example. The figure shows the time in seconds along the x-axis as a time scale, arranges the components above the time scale, and received observations are shown as arrows pointing to a specific time on the time scale. The use case in this example is configured to send observations at a regular interval of 30 seconds. With regards to our running example, we selected the minimum time interval of 30 seconds, as the existing entities (vehicles) need some time to continue driving and produce meaningful observation data. Each incoming observation triggers the *Coordination* that decides which other components are required at that time. At the beginning of the framework execution, the *Coordination* stores received observations and forwards them to the *Situation Detection*. However, since there is not enough data, the *Situation Detection* does not provide a situation and the *Coordination* applies the fallback rules. Once there is enough data (at second 600), the *Situation Detection* returns a specific situation ID. Then, the *Parameter Optimization* optimizes the parameters for the first time. Strategy Selection is omitted at this point, because we decided to first optimize the parameters of the current strategy to see if the performance of the strategy can be sufficiently improved by an optimized parameter setting. In the presented example, the number of optimization attempts per situation is set to five. Thus, after 3,600 seconds execution time, the *Coordination* has already triggered five optimization attempts and now additionally triggers the *Strategy Selection*. This results in the selected strategy being executed for at least one hour and optimized several times before a new selection is made, which allows the running example to perform adaptations and observe performance changes.

6 DOMAIN DATA MODEL

The *DDM* is a representation of the use case for the framework and serves as configuration file enabling the generic applicability of the framework. This means that these settings strongly depend on the chosen use case and can individually be enriched by use-case-specific parameters. The *DDM* is defined using YAML, as it is easy to read for humans and can be used even without programming knowledge. Therefore, it is well suited for the domain expert and provides separation of concerns. The *DDM* consists of four main parts: (i) use case, (ii) context, (iii) parameter_options, and (iv) performance_measures. In the following, we quickly describe each of these parts as the details are of technical nature. The interested reader can find an extensive description of the *DDM* in our technical report [37].

Use Case Information: The first part of the *DDM* is called *use case*, which contains general information about the use case. It contains the identifier *name* and a list of available adaptation planning strategies called *available_strategies*. The *Strategy Selection* component of the framework uses this list to determine the most promising strategy for the current situation. Finally, it contains the *fallback_rules* containing a path to a Python file that defines fallback rules for the framework that will be used whenever the *Situation Detection* is not possible.

Context: The second part of the *DDM* is called *context* and specifies the context *data*, i.e., observations, the use case sends to the framework. Furthermore, this part defines the configuration of the *Situation Detection* component with the key *situation_detection_settings*. The data key contains any number of context parameters from the use case with unique name-based identifiers and a *data_type* specification (e.g., `int` and `double`). The situation_detection_settings key consists of the two keys *algorithm* and *settings*. The algorithm key expects the definition of an available situation detection algorithm. So far, four algorithms are available that can be easily extended in the future. We describe them as well as their additional configuration parameters in more detail in Section 7: `RuleBased`, `K-Means`, `DBSCAN`, and `OPTICS`.

Parameter Options: The third part of the *DDM* is called *parameter_options*. It defines tunable input parameters of the strategy and provides configuration information for the *Strategy Selection* component. This part consists of the *options* for the input parameters and the *strategy_selection_settings*. The options key contains an arbitrary number of input parameter options for strategies defined using a *data_type*, *min* and *max* values, and an optional list of relevant *strategies*. The *strategy_selection_settings* key consists of five mandatory keys: *observations_between_adaptations*, *min_optimization_attempts*, *window_size*, *threshold_exceeds*, and *method* and one optional key called *hypervolume_threshold*. For a detailed explanation of these keys, please refer to Section 8.

Performance Measures: Finally, the last part of the *DDM* is called *performance_measures* and defines indicators of the performance of the defined use case. This part contains any number of performance measures from the use case, with unique names. Each performance measure consists of three mandatory keys *data_type*, *higher_is_better*, and *reference_value*, and an optional key called *threshold_value*. Again, a detailed explanation of these keys can be found in Section 8.

7 SITUATION DETECTION COMPONENT

The *Situation Detection* component is responsible for identifying the current situation the managed system of the use case is currently experiencing, as depicted in Figure 2. So far, this component provides four methods: (i) rule-based, (ii) K-Means, (iii) DBSCAN, and (iv) OPTICS, which can be easily extended. We selected these four methods to provide an opportunity to integrate domain-knowledge using the rule-based method and three methods that do not require any domain knowledge and operate unsupervised. We selected K-Means as a well-known clustering technique that can be useful when the number of different situations is known in advance. Further, we select DBSCAN and OPTICS as clustering techniques that require less parameters and, hence, reduce the preparation and parametrization tasks to a minimum. All methods operate on all context data available in the system. The *Situation Detection* component computes the current situation and returns a situation ID to the *Coordination* component.

The situation detection process can be defined as a mathematical function mapping observation data from the use case to an integer value. This value represents the situation ID as defined in Equation (1) with a value range $[-1, \infty)$, where the value -1 indicates that the situation could not be detected. This could be due to: (i) insufficient amount of observation data, (ii) noisy observation data. A classification as noise could indicate a novel situation, or measurement inaccuracies in the use case. In the case that the *Situation Detection* classified the current situation as -1, the framework does not invoke any other components but applies user-defined fallback rules. If the returned situation ID is equal to or greater than zero, then the *Situation Detection* component has determined a valid situation and the *Strategy Selection* and *Parameter Optimization* can be invoked. The actual value of the situation ID does not allow for further interpretation regarding the similarity of situations. As simplified example, let us assume that the component identified three situations $s_1 = 0, s_2 = 1, s_3 = 10$. This means that these three situations exist and are all different from each other. Moreover, the proximity of the values 0 and 1 does not mean that the situations s_1 and s_2 are more similar to each other than the situation s_3.

$$sit\_det(context) = \begin{cases} -1, & \textit{if} \text{ situation is classified as noise} \\ >= 0, & \text{otherwise} \end{cases} \tag{1}$$

Since the use case regularly sends new observations, the amount of data grows consistently and might result in distinct assignment to situations during operation of the component. This means the situations identified during the last situation detection process may not be the same as those identified in the current process. Thus, the *Situation Detection* component updates its learned models after each execution to match the latest findings to the observation data. Due to the permanent monitoring of the framework, the amount of observation data will grow over time. At the moment, the clustering techniques of the situation detection component use all available data for identifying the situation. In terms of the rule-based situation detection, only the latest observation is used. Since the clustering techniques use all available data, the number of observation points grows in time and a mechanism should be integrated to prune too old or irrelevant data. This should decrease the time to result of the situation detection and avoid getting stuck in too-old situations.

We provide two types of situation detection mechanisms, one rule-based mechanism and three clustering algorithms that can be selected and configured by the user in the *DDM*. However, the component is not limited to these four techniques and can be extended easily with further or use-case-specific situation detection techniques due to its modular structure. The component receives the *DDM* and all existing observations and selects the configured algorithm for the *Situation Detection*. In all cases, the component retrieves required parameters for the selected technique from the *DDM* and invokes the configured technique. All techniques return the `situationIDs` for all observations, that is, the cluster to which each observation in the dataset is assigned. The component then updates its situation model of all observed data with the latest classification and returns the `situationID` of the new observation to the *Coordination* component.

The rule-based situation detection offers the possibility to integrate domain knowledge in the identification process of this component. For example, in the platooning use case, the user could specify frequent traffic volumes for which they know the best-performing configuration of the adaptation planning system. The user defines the rules in form of a Python file that is loaded and executed by the component. As long as the user provides a script that matches our definition in

Equation (1), this Python file could contain arbitrary complex operations. Further, the user could adapt the given rules and include new domain knowledge gained from the framework operation. In the context of this article, we omit updating the user-provided rule set with new knowledge from previous executions, but this could be valid future work following existing approaches such as References [10, 19, 47].

In addition to the static rule-based situation detection, we provide three clustering-based situation detection methods. Due to their unsupervised learning methods, they can automatically detect new situations and do not require domain knowledge [5, 17]. The first approach is k-means with a predefined parameter k, or alternatively in combination with *gap statistics* [61] that automatically selects the parameter k. When using gap statistics, the user needs to specify a minimum and maximum value for k, but no further user interaction is required. Since the performance of k-means heavily depends on k and is not able to identify noise, we additionally integrate two density-based clustering approaches. Therefore, we select DBSCAN and OPTICS, which do not require a number of clusters as input. Instead, DBSCAN requires the definition of `min_samples` and ϵ (`eps`) for which domain knowledge from the user is required. OPTICS needs the parameters `min_samples` and `min_cluster_size`, which can be determined by considering how long a situation is usually active in the use case and how many observations are sent to the framework. Both density-based clustering algorithms can classify observations as noise, which could happen when the use case observes a new situation for a short time.

8 STRATEGY SELECTION COMPONENT

The *Strategy Selection* is the second component invoked by the *Coordination* component and is responsible for selecting the most promising adaptation planning strategy. This functionality is based on the No-Free-Lunch Theorem for optimizations [65] and the identified situation-dependent behavior of adaptation planning strategies [40]. To do this, the framework uses experience gained from previous executions of the strategies in similar situations. However, which algorithm performs best in a new situation is not known *a priori*. Therefore, the component tests available strategies and starts a new round of learning for that situation. A general definition of the algorithm selection problem can be found in Reference [55]. In the following, we explain the workflow of the *Strategy Selection* and refer to Algorithm 2.

Similar to the *Situation Detection*, this component also receives the *DDM* as input. Additionally, it receives the currently active adaptation planning strategy, the number of optimization attempts already performed for this strategy, and all available observations for the current situation containing the performance measures of the strategy. First, the *Strategy Selection* sets the currently active strategy as the selected strategy (Line 1). Then, it checks that enough optimization attempts have been made to decide whether the strategy should be changed (Line 2). If the actual number of optimization attempts has not reached the minimum number of optimization attempts, then it means that the *Parameter Optimization* component might need more time to optimize the parameters of this strategy, and this component returns the currently active strategy (Line 3). If the required number of optimization attempts has already been reached (Line 4), then this component can select another strategy if the current strategy does not meet the performance expectations (Lines 5–8). Therefore, the component analyzes the performance of the strategy in the last observations with respect to a defined threshold and counts the number of times the threshold is exceeded within a defined *window_size*. The component provides two ways to define this threshold (as explained later in this section): (i) hypervolume threshold and (ii) individual value thresholds. Afterwards, it checks whether this number is above the predefined maximum allowed threshold violations (Line 9). If a new strategy should be selected, then it checks whether all strategies were already executed for this situation and selects the one yielding the highest average Hypervolume

ALGORITHM 2: Pseudocode workflow of the Strategy Selection component.

```
   Input: DDM, current strategy, number of optimization attempts already performed, all observations for
          the current situation
1  strategy ← current strategy;
2  if number of optimization attempts < DDM.min_optimization_attempts then
3      return strategy;
4  else
5      exceed_counter ← 0;
6      for observation within DDM.window_size do
7          if thresholds exceeded then
8              exceed_counter++;
9      if exceed_counter >= DDM.threshold_exceeds then
10         if all strategies already executed for this situation then
11             strategy ← best-performing strategy in history;
12         else
13             strategy ← next strategy determined in DDM;
14 return strategy;
```

of performance measurements (Lines 10–11). Otherwise, if at least one strategy was not executed for this situation, then the *Strategy Selection* retrieves the next one from the *DDM* (Lines 12–13). This can be seen as a trial-and-error phase, since the decision cannot be based on experience and the component is forced to try new combinations. Finally, the component returns the selected strategy to the *Coordination* component (Line 14).

The *Strategy Selection* component provides two possibilities to determine whether an algorithm meets the expected performance or should be modified. The first method the component offers is the Hypervolume threshold method, which reduces the performance measures to a single score. To calculate the Hypervolume, the user must specify reference values for each performance measure in the *DDM*. However, the downside of this method is that it weights measures with a larger value range more heavily, so the user should apply a normalization mechanism before sending the performance measures to the framework. Still, the advantage of this method is that the performance of the overall adaptation planning system is condensed into one metric and the user only needs to specify one threshold value. The second method is to set individual value thresholds for each performance measure of the *DDM*. Whenever one of the performance measures exceeds its threshold, the *Strategy Selection* component counts this as a violation, regardless of any possibly perfect performance of the other measures. This method allows the user to have more impact on the individual performance measures and value ranges of these measures are less important. Additionally, the user can easily extent the functionality of this component due to its modular design. For instance, Machine Learning techniques such as Random Forests [21] can be integrated to learn a model for the *Strategy Selection*.

9 PARAMETER OPTIMIZATION COMPONENT

The last component is the *Parameter Optimization* component, which is invoked when a new strategy is determined, the situation changes, or the performance of the strategy decreases. This component uses Bayesian Optimization, which performed best in our preliminary study [40] to determine the best-performing parameter setting for the selected strategy. Therefore, it uses historical observation data of the same situation and strategy combination. If the situation-strategy combination has not changed since the last invocation of this component, then the Bayesian Optimization integrates only the last observation into the optimization model to compute new parameters. If either the situation or the selected strategy has changed since the last invocation, then the optimization model must be re-trained using historical data of the new situation-strategy combination, if available. This allows the *Parameter Optimization* to react to the current situation and strategy and learn from previous decisions. The *Parameter Optimization* component returns the new parameter set for the strategy to the *Coordination* component, which forwards the adaptations to the use case.

10 EVALUATION

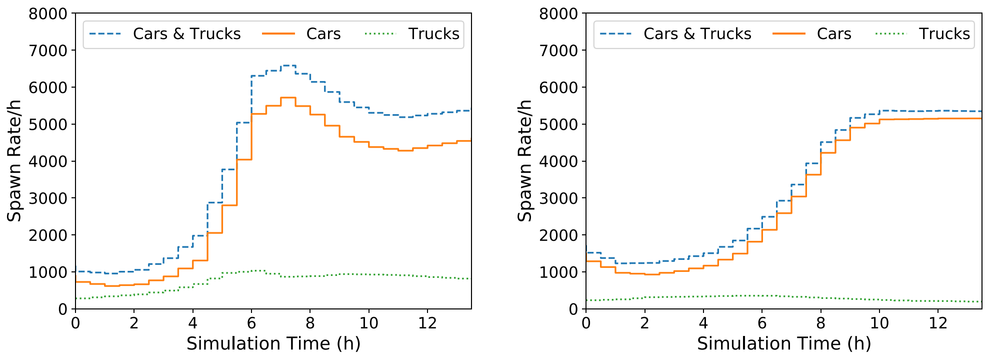
In this section, we evaluate the proposed self-aware optimization framework. Since our framework is a novel combination of approaches and there is no mechanism that incorporates situation detection, algorithm selection and parameter optimization into one approach, we cannot compare our framework to state-of-the-art methods. Hence, we focus on a feasibility study in this work and plan an in-depth performance evaluation of all components isolated against state-of-the-art mechanisms in the future. Therefore, Section 10.1 summarizes the methodology of our evaluation. Sections 10.2, 10.3, and 10.4 evaluate the Situation Detection, Strategy Selection, and Parameter Optimization Component, respectively. Afterwards, Section 10.5 analyzes the overall performance of the entire framework, and Section 10.7 discusses threats to the validity of the evaluation.

10.1 Methodology

In this work, we use the platooning coordination use case as a running example of our self-aware optimization framework. We first define the applied scenarios, summarize the testbed, and specify the framework configuration before proposing our baseline approaches.

Scenarios: We use a simulated road section of the German highway A8, which ranges from the Stuttgart interchange to the Stuttgart-Degerloch exit. According to Süddeutsche Zeitung, this section is one of the busiest highway sections in Germany [14]. In addition to the realistic model of this highway section, we use real traffic data provided by the Federal Highway Research Institute of Germany [1] to define the vehicle spawn rates for our simulation. After a detailed analysis of the traffic values for each day of the week, we selected Wednesday as the representative weekday and Saturday as the representative weekend day. Figure 7 shows the traffic volume for the selected days between 12:00 AM and 2:00 PM. As the simulation of such high traffic volume requires high computational power and shows long computation time, we decided to only simulate the first 14 hours of a day. This time interval contains a typical traffic volume profile (including a nightly low traffic volume, the first rush hour of a day, and the increasing traffic volume of a second rush hour) for weekdays as well as weekends and, therefore, provides a good balance between long runtime and comprehensive simulation. We set the platooning percentage of all vehicles to 70%, as we assume that not every vehicle is capable of platooning or drivers choose not to participate. Furthermore, we set the maximum speed limit of cars to 120 km/h, which corresponds to the actual speed limits on this section [7]. In our evaluation, we use two types of situation detection (OPTICS and rule-based situation detection) and two types of triggers for strategy selection (Hypervolume- and threshold-based triggers), which results in four simulations per traffic profile. Since our approach involves Bayesian Optimization that incorporates randomness, we run three different random seeds in the traffic simulator SUMO for each simulation.

Testbed: We perform our simulations in the cloud of the Chair of Computer Science II at the University of Würzburg. This cloud consists of 18 hosts, each running RHEL-7-8.2003.0.el7.centos and oVirt Node 4.3.10 with KVM version 2.12.0. The cloud contains one large ProLiant DL380 Gen9 host with two Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz CPU sockets and eight cores per



(a) Wednesday (b) Saturday

Fig. 7. Considered traffic scenarios of the framework evaluation for Wednesday on the left and Saturday on the right. Total number of spawning vehicles is depicted as blue dashed line, cars are depicted as solid orange line, and trucks are depicted as dotted green line.

socket. The remaining hosts are ProLiant DL160 Gen9 type with two CPU sockets of type Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz, eight cores per socket, and two CPU threads per core. We use three identical virtual machines for the simulations, which are deployed in our private cloud. Each virtual machine has two CPU sockets, each with 4 cores running at 2.6 GHz and 32 GB available RAM. We measure the simulation runtime of our scenarios, resulting in an average runtime of 9.5 days for the Wednesday scenarios and 9 days for Saturdays, which is due to a lower traffic volume on Saturday. Since our goal for this article is a feasibility study, we do not measure and report any more performance metrics besides the overall runtime. Still, in the future an in-depth performance analysis is planned that incorporates detailed measurements for all components.

Framework Configuration: As data input for the situation detection, we use the amount of vehicles on the road. We defined the rules for the rule-based situation detection according to the definitions for peak hours, medium, and low traffic volumes from the German city of Rostock [2], which also includes traffic volumes of highways around the city: We consider low, medium, and high traffic situation where the maximum number of vehicles on the road section is 120, between 121 and 280, and above 280 vehicles, respectively. OPTICS requires the definition of the minimum number of points and the minimum cluster size, both of which we set to a value of 45, which we derived in a preliminary parameter study.

Similar to the situation detection, we also evaluate two triggers for the strategy selection component: Hypervolume and individual thresholds. Both methods incorporate the four objective metrics to assess the performance of the currently active strategy [57]: (i) throughput, (ii) time loss, (iii) platoon utilization, and (iv) platoon time. The Hypervolume requires the definition of a reference value outside the range value of the metrics, which we set to −0.1. We set the Hypervolume threshold to 0.3 and consider a time window size of five, in which the Hypervolume must fall below the threshold at least three times to trigger the strategy selection. In line with our preliminary study [37, 40], we set the individual thresholds to: throughput = 0.5, time loss = 0.9, utilization = 0.62, and platoon time = 0.3. We set these values to find a tradeoff between sensitive responses to degrading performance metrics and avoiding jitter. Further, we define the initial trial phase for the strategy selection to 10 optimization cycles and specify the order in which the platooning coordination strategies are selected: Best-Distance, Best-Velocity, as well as Best-Distance-and-Lane. The Best-Distance strategy analyzes the distance between vehicle and possible platoons and selects the platoon with the lowest longitudinal distance. The Best-Velocity strategy defines the best

Table 1. Configuration of the Framework and Tested Strategies, Algorithms, and Methods Used in the Evaluation

DDM Part	Parameter	Value
Use Case	Available strategies	Best-Distance, Best-Velocity, Best-Distance-and-Lane
Situation Detection	Algorithm	RuleBased, OPTICS
Strategy Selection	Method	Hypervolume, threshold
	Min. opt. attempts	10
Hypervolume	Reference values	−0.10
	Threshold	0.30
	Time window size	5
	Threshold exceeds	3
Thresholds	Throughput	0.50
	Time loss	0.90
	Platoon utilization	0.62
	Platoon time	0.30

Q6

Table 2. Configurations of the Baseline Approaches Used in the Evaluation

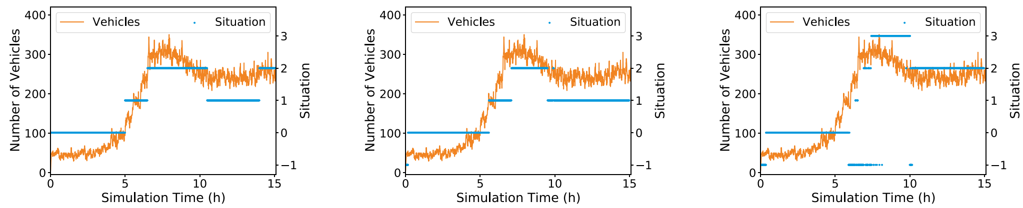
Parameter Name	Best-Distance	Best-Velocity	Rules I	Rules II
Advertising duration [m]	10	10	10	5
Search distance front [m]	-	600	600	400
Search distance back [m]	-	250	250	200
Max. speed difference [km/h]	35	-	-	-
Speed threshold lane 2 [km/h]	100	100	100	100
Speed threshold lane 3 [km/h]	130	130	130	130
Speed threshold lane 4 [km/h]	160	160	160	160

matching platoon by calculating the velocity difference between platoon and vehicle and selecting the platoon with the lowest positive speed delta. The Best-Distance-and-Lane strategy not only calculates the longitudinal distance of vehicle and platoon but penalizes the number of lanes between them.

To evaluate the performance of our framework against a set of baseline approaches, we apply the Best-Distance, Best-Velocity, and a rule-based strategy to the two scenarios. According to our previous study [40], these two strategies performed best and should be the strongest competitors. We design the rule-based strategy as gold standard strategy in which we combine the knowledge from the previous study into if-then-else rules to analyze how well our self-aware framework performs compared to the optimum. Table 2 summarizes the configurations of our baseline strategies in line with our previous study [40]. The rule-based strategy applies the Best-Velocity strategy with two configurations dependent on the number of vehicles and average car speed. It applies the first configuration if the number of vehicles is below 500 and the car speed is above 125 km/h and the second configuration otherwise. We also apply the same set of rules as fallback-mechanism in our framework when the applied situation detection cannot detect the current situation.

10.2 Evaluation of the Situation Detection Component

In line with the workflow of our optimization framework, we start our evaluation with the situation detection component and analyze how well the implemented situation detection approaches actually identify existing situations and their changes. Keep in mind that we currently only want to analyze the feasibility of the proposed framework and its components and explicitly exclude a

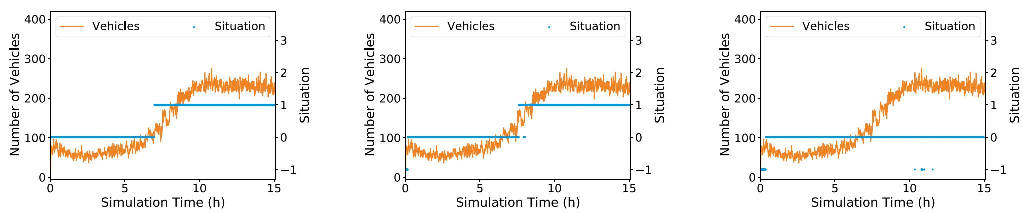


(a) Ground truth for the situation detection.

(b) Detected situations when applying rule-based situation detection.

(c) Detected situations when applying OPTICS situation detection.

Fig. 8. Actual situations of the ground truth and detected situations of the rule-based and OPTICS approach for Wednesday traffic data. The orange line represents the vehicle spawn rate at a specific point in time. The blue dots represent the detected situation at the current point in time incorporating all previously observed data points.



(a) Ground truth for the situation detection.

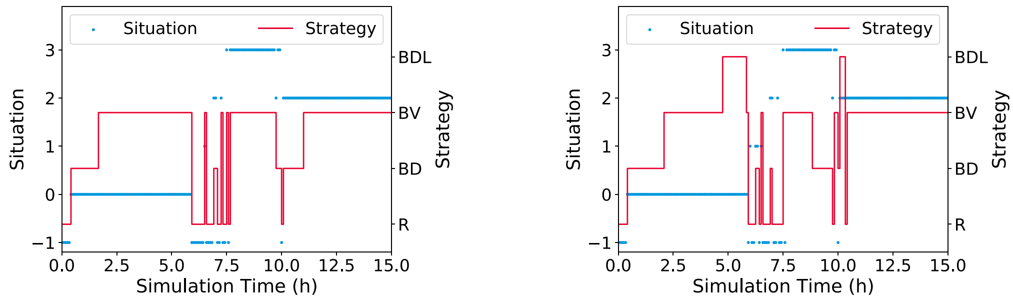
(b) Detected situations when applying rule-based situation detection.

(c) Detected situations when applying OPTICS situation detection.

Fig. 9. Actual situations of the ground truth and detected situations of the rule-based and OPTICS approach for Saturday traffic data. The orange line represents the vehicle spawn rate at a specific point in time.

performance analysis of all components and the framework as a whole. This also excludes details computation time measurements. This component uses the current amount of vehicles on the road to identify a situation. Therefore, we analyze the detected situations during the simulation for both scenarios and compare the rule-based and OPTICS approaches to the ground truth. The ground truth uses the definitions of peak hours, medium, and low traffic volumes as described earlier. Figure 8 shows the ground truth for situation detection and the results of the component applied to the Wednesday scenario. The orange line represents the vehicle spawn rate, while the blue dots represent the cluster ID, that is, the detected situation, at a given time. The figure shows the cluster numbers assigned when the observation first occurred representing the situation based on which the framework makes its decisions. When comparing the identified clusters in Figures 8(a) and 8(b) it can be seen that the rule-based situation detection component is close to ground truth, as it identifies all three situations, but assigns fewer observations to the peak traffic cluster. In addition, the rule-based approach does not detect the start of the second peak traffic cluster. The good performance of this approach was expected, since the rules were derived from the ground truth. The situation detection using OPTICS, as shown in Figure 8(c), identifies the situations using clustering mechanisms and identifies four different situations but considers some observations as noise. The four identified situations are less evenly distributed in terms of the number of observations they contain compared to the ground truth, as the length of the resulting blue bars strongly vary. Nevertheless, this mechanism is able to distinguish different situations as seen in the different height levels of the resulting blue lines even if they are not completely consistent with the ground truth.

The results of the situation detection component applied to the Saturday scenario are depicted in Figure 9. Again, the orange line represents the vehicle spawn rate, and the blue dots represent



(a) Selected Strategies when using the OPTICS situation detection and Hypervolume trigger.

(b) Selected Strategies when using the OPTICS situation detection and individual threshold triggers.

Fig. 10. Strategy selection on Wednesday traffic data. Blue points represent the detected situation at a specific point in time. The red line represents the selected adaptation planning strategy at a specific point in time (R = *Rules*, BD = *BestDistance*, BV = *BestVelocity*, and BDL = *BestDistanceAndLane*).

the identified cluster ID. While the ground truth and rule-based approach show two identified situations with a switch at around 7.5 hours, the OPTICS situation detection only shows one blue line with some outliers after 10 hours. Hence, similarly to the Wednesday scenario, the rule-based approach is close to the ground truth, which is not surprising, since the rules were derived from it. However, the OPTICS approach shows a different behavior, as it is not able to identify at least two different situations and clusters all observations into one situation. The poor performance of this approach could be due to an unfavorable parameter configuration resulting from our preliminary parameter study. Another factor could be the lower number of vehicles on the road compared to the Wednesday scenario, which could lead to very similar observation data. Further evaluation using more extensive scenarios and additional parameter studies may provide more insight in the future.

In summary, this evaluation shows that the rule-based approach performs well against the defined ground truth for both scenarios. The OPTICS approach identifies distinct situations in the Wednesday scenario, but only a single situation for the Saturday scenario. The ground truth derived rules work well but are a very rigid approach and do not provide flexibility for future changes. A rule set must be defined at design time using expert knowledge and will not be further adapted. However, the clustering approach OPTICS provides more flexibility but does not find the situations defined in the ground truth as reliably. In the future, extended simulations with, for example, several days, could reveal more potential for improvements. In addition, rule learning methods could be used to adapt the rule-based situation detection during runtime.

10.3 Evaluation of the Strategy Selection Component

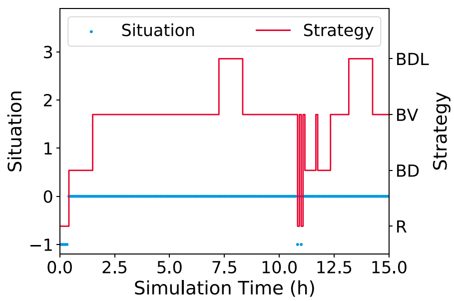
In this section, we analyze the proper operation of the strategy selection component. We analyze how a change in the identified situation affects the choice of strategy by presenting the selected strategies in combination with the identified situation over time. Keep in mind that we currently only aim at analyzing the feasibility of the proposed framework and its components and explicitly exclude a performance analysis of all components and the framework as a whole. This also excludes details computation time measurements. Therefore, Figure 10 shows the selected strategies for the Wednesday scenario using OPTICS as the situation detection mechanism and the Hypervolume trigger in Figure 10(a) as well as the individual thresholds as trigger in Figure 10(b). We decided to use continuous line charts with vertical lines representing a strategy change to better visualize the changed strategies especially in cases where the selection changes back and forth frequently.

We base this evaluation solely on OPTICS, as it identifies different situations for the Wednesday scenario and is able to handle new situations not defined in a rule set.

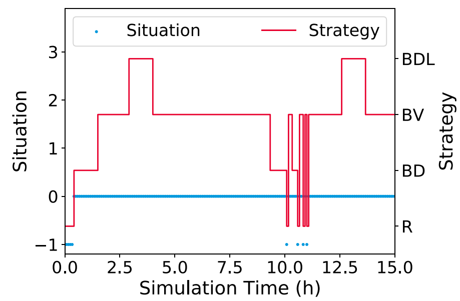
The blue points represent the determined situation, while the red line illustrates the selected strategy at a certain point in time, that is, the height of the line represents the selected strategy. The left figure shows that the strategy selection component selects a strategy and switches to the next one if the performance metrics fall below the thresholds and the triggers activate the selection. When using the Hypervolume trigger, the strategy selection remains at the Best-Velocity and does not switch to the Best-Distance-and-Lane within the first six simulation hours compared to the individual threshold trigger. After this time, the observations are classified as noise by the situation detection, which causes the strategy selection to revert to the rule-based strategy. Whenever new situations occur, the strategy selection starts with the Best-Distance strategy and tests its performance before switching to the Best-Velocity strategy. The results show that the individual thresholds trigger the strategy selection more often compared to the Hypervolume trigger, as the selection component examines the Best-Distance-and-Lane twice. In summary, the strategy testing phase at the beginning of new situations, the stabilization to well-performing strategy and the fallback to rules is the intended behavior of the framework and tells us that it is working properly. However, since the individual thresholds trigger the strategy selection more often, this may indicate that the individual thresholds are too restrictive and could be relaxed to avoid jitters between strategies.

Figure 11 shows the results of the strategy selection component for the Saturday scenario using OPTICS and rule-based situation detection in combination with the Hypervolume and individual threshold triggers. The reason for using the rule-based situation detection in this evaluation is that OPTICS situation detection was not able to identify more than one situation for the Saturday scenario. Figure 11(a) presents the OPTICS and Hypervolume evaluation, Figure 11(b) presents the OPTICS and individual threshold evaluation, Figure 11(c) illustrates the rule-based and Hypervolume evaluation, and Figure 11(d) shows the rule-based and individual threshold evaluation. Again, the blue points represent the identified situation, and the red line represents the selected strategy at a given time. All figures show the desired exploratory behavior of the strategy selection when a new situation occurs due to the step-wise strategy change at the beginning. If a strategy performs well, then it is not replaced and remains active until the triggers indicate a performance degradation. Since the OPTICS situation detection identifies only one situation and classifies some observations as noise, it shows a clear step-wise strategy change and a reversion to the rule-based strategy when the situation detection reveals noise. When using the rule-based situation detection, the strategy selection is more stable, since no fallback mechanisms are required. However, Figure 11(c) shows an anomaly in the strategy selection behavior, as the detection of a new situation does not trigger a new exploration of strategies after around eight hours. A detailed analysis of this behavior led us to the conclusion that the detection of a situation change was not perfectly aligned with the strategy selection component and, hence, resulted in a lost situation change. Thus, the currently active strategy, that is, the Best-Velocity, remains active until about 11 hours of simulation time. At this point, the Hypervolume trigger indicates a performance degradation of the current strategy and the strategy selection selects the Best-Distance strategy. However, it is discarded after the initial trial period and the strategy selection switches to the Best-Distance-and-Lane strategy. The same lost update of a new situation can be observed in Figure 11(d). However, this figure shows a faster discarding of the currently active strategy, similar to the behavior in Figure 11(b). This also indicates that the individual thresholds might be too restrictive and could be relaxed in the future to produce a more stable result.

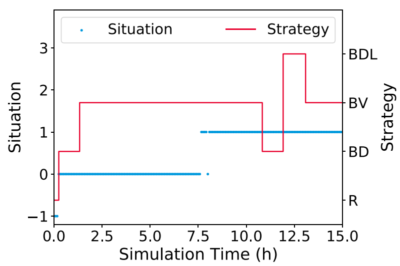
In summary, this evaluation shows that both algorithm selection trigger methods work properly and activate the algorithm selection when the performance of the currently active strategy



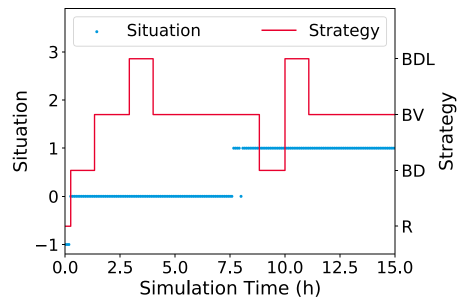
(a) Selected Strategies when using the OPTICS situation detection and Hypervolume trigger.



(b) Selected Strategies when using the OPTICS situation detection and individual threshold triggers.



(c) Selected Strategies when using the rule-based situation detection and Hypervolume trigger.



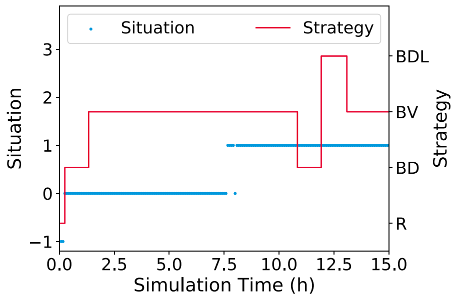
(d) Selected Strategies when using the rule-based situation detection and individual threshold triggers.

Fig. 11. Strategy selection on Saturday traffic data. Blue points represent the detected situation at a specific point in time. The red line represents the selected adaptation planning strategy at a specific point in time (R = *Rules*, BD = *BestDistance*, BV = *BestVelocity*, and BDL = *BestDistanceAndLane*).

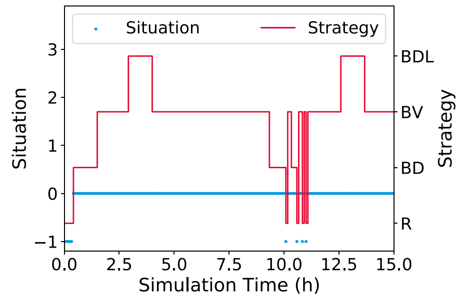
deteriorates. While the Hypervolume threshold provides a more stable result, the individual thresholds appear to detect performance degradation earlier. Therefore, the individual thresholds explore more possible strategies, but also result in higher jitter compared to the Hypervolume. However, the definition of the individual thresholds can be adjusted in future evaluation studies to achieve a tradeoff between detecting performance degradation quickly and reducing jitter. All in all, both methods work properly and are capable of triggering the algorithm selection.

10.4 Evaluation of the Parameter Optimization Component

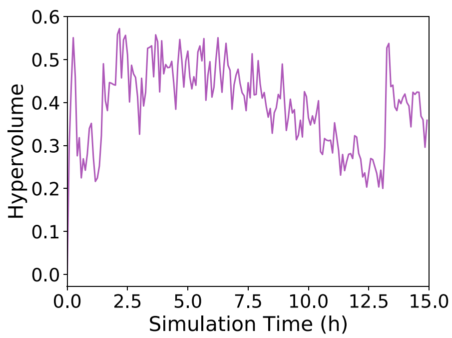
We evaluate our optimization component by analyzing the course of the Hypervolume metric used by this component to optimize the parameter configuration of the current adaptation planning strategy. Keep in mind that we currently only want to analyze the feasibility of the proposed framework and its components and explicitly exclude a performance analysis of all components and the framework as a whole. This also excludes details computation time measurements. The used Hypervolume metric (cf. Reference [63]) accumulates the platooning metrics into one objective metric that can be used by the single-objective Bayesian Optimization. Figure 12 shows evaluations of the Saturday scenario using rule-based situation detection and Hypervolume as trigger for the strategy selection component on the left (Figures 12(a) and 12(c)). The right side of the figure shows measurements for the Saturday scenario using OPTICS as situation detection mechanism and individual thresholds as triggers for strategy selection (Figures 12(b)



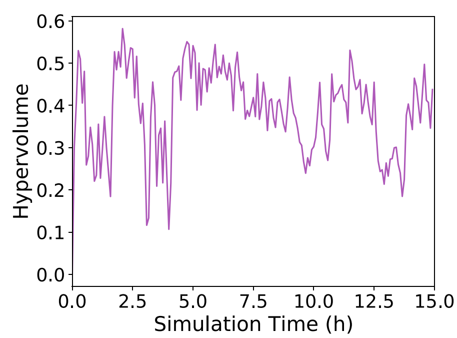
(a) Selected Strategies when using the rule-based situation detection and Hypervolume trigger.



(b) Selected Strategies when using the OPTICS situation detection and individual threshold triggers.



(c) Hypervolume score of the selected strategy when using the rule-based situation detection and Hypervolume trigger.



(d) Hypervolume score of the selected strategy when using the OPTICS situation detection and individual threshold triggers.

Fig. 12. Evaluation of the optimization component on the Saturday scenario. The left side represents configurations using the rule-based situation detection and Hypervolume triggers. The right side illustrates OPTICS situation detection and individual threshold triggers (R = *Rules*, BD = *BestDistance*, BV = *BestVelocity*, and BDL = *BestDistanceAndLane*).

and 12(d)). The top figures show the identified situations in blue in combination with the selected strategies in red. The lower figures summarize the course of the Hypervolume metric, that is, the performance indicator of the platooning coordination strategy. The course of the Hypervolume metric appears to be very fluctuating for both configurations during the simulation time. This was expected behavior, since the optimization component needs some time to learn which parameter setting works well for which strategy and situation. Therefore, it makes most sense to analyze time windows of the Hypervolume progression where the identified situation and strategy remain stable. This is also a reason for choosing Saturday scenarios for this evaluation, as traffic volumes do not fluctuate as much as in Wednesday scenarios, which allows for longer time frames per situation and strategy. When analyzing the first stable phase on the left between 2.5 and 7.5 hours of simulation time, the Hypervolume starts with a value of about 0.5 Hypervolume points and drops to 0.3 Hypervolume points. Then, it stabilizes back to about 0.5 Hypervolume points, indicating that the optimization component has explored different parameter settings and stabilized to a well-performing set of parameters. As discussed earlier, the change in the situation is lost at about 7.5 hours of simulation time, resulting in a sharply decreasing trend in the Hypervolume. This leads to the extended Hypervolume threshold that triggers the strategy selection at about 11 hours of simulation time. The other configuration, depicted on the right, captures OPTICS and individual thresholds. In this evaluation, we can analyze the Hypervolume score for the simulation period starting at 4 hours up to 8 hours of simulation time. The Hypervolume score shown on the bottom right starts at a low value of around 0.2 score points, but quickly increases to a value of 0.4 score points. This low start value is due to the recent strategy change from the Best-Distance-and-Lane strategy, which was discarded in favor of the Best-Velocity strategy after its initial trial phase. After that, the Hypervolume score shows a slight increase to a value of about 0.58 score points, but then decreases again to values between 0.4 and 0.5 score points. This indicates that the Optimization component finds better parameter settings for the selected strategy and then explores new parameter settings that unfortunately lead to worse Hypervolume values. This triggers the strategy selection, and, since all existing strategies have already been explored, the best-performing strategy will be selected even if it again triggers strategy selection and parameter optimization.

In summary, this evaluation shows us that the Optimization component has the potential to optimize the parameter settings of the adaptation planning strategies, as the Hypervolume score remains stable and shows slight increases in stable performance for situation and selected strategy. However, negative effects also occur when the Optimization component explores new parameter settings, which may lead to worse results compared to the previous settings that performed well. This indicates that the stable phases of identified situations and selected strategies, that is, the time for the Optimization component to optimize the parameter settings, may be too short to find stable configurations with good performance. Extended evaluations over several days or even weeks could provide more insight into the required amount of experience for the Optimization component and increase the overall performance of this component.

10.5 Evaluation of the Entire Framework

In our final evaluation, we analyze the overall functionality of the framework and perform an integrative evaluation using all components at the same time. Keep in mind that we currently only want to analyze the feasibility of the proposed framework and its components and explicitly exclude a performance analysis against state-of-the-art approaches of all components and the framework as a whole. This also excludes details computation time measurements. First, we compare the four defined configurations of the framework with the three baselines in terms of the four platooning metrics of throughput, time loss, platoon utilization, and platoon time. Table 3 presents the mean and standard deviation results for these metrics for the Wednesday scenario and Table 4 summarizes the results for the Saturday scenario for the three repetitions. We highlight the best values of each platooning metric for the baseline group and the framework group in bold. In both evaluation scenarios, the throughput metric results for all baselines and framework configurations are very close, with values between 0.9943 and 0.9952 and low standard deviations. In the Wednesday scenario, the Best-Distance baseline and rule-based situation detection combined with Hypervolume thresholds perform best on the throughput metric with values of 0.9952 and 0.9946, respectively. In the Saturday scenario, all configurations of the framework perform equally well, while the Best-Velocity baseline performs best on the throughput metric with values of 0.9950 and 0.9951, respectively. All applied configurations and baselines show higher diversity for the time loss metric, ranging from 0.8992 to 0.9122 for Wednesday and from 0.9255 to 0.9411 for Saturday. Rule-based situation detection combined with individual thresholds performs best for this metric among all configurations tested, with a value of 0.9122 and 0.9333, but achieves a lower value compared to the Best-Velocity baseline, with a value of 0.9199 and 0.9411 for Wednesday and Saturday, respectively. Results for the platoon utilization metric range from 0.6251 to 0.7176 and from 0.5999 to 0.7101 for Wednesday and Saturday, respectively. For this metric, the fallback rule baseline among the baselines and the OPTICS situation detection in combination with Hypervolume and individual thresholds performs best. Finally, the results for the platoon time metric range

Table 3. Evaluation Summary of the Average and Standard Deviation for Performance Metrics Throughput, Time Loss, Platoon Utilization, and Platoon Time for the Wednesday Scenario

Configuration	Throughput		Time Loss		Platoon Utilization		Platoon Time	
	mean	std	mean	std	mean	std	mean	std
Best Distance	**0.9952**	0.0	0.8992	0.0	0.6251	0.0	0.4908	0.0
Best Velocity	0.9942	0.0	**0.9199**	0.0	0.6973	0.0	0.6109	0.0
Fallback Rules	0.9950	0.0	0.9198	0.0	**0.7176**	0.0	**0.6518**	0.0
OPTICS & Hv	0.9943	0.0003	**0.9122**	0.0022	**0.6690**	0.0030	**0.5442**	0.0090
Rule-based & Hv	**0.9946**	0.0004	0.9102	0.0011	0.6647	0.0039	0.5302	0.0076
OPTICS & Th	0.9945	0.0003	0.9110	0.0014	0.6566	0.0072	0.5275	0.0119
Rule-based & Th	0.9943	0.0003	0.9108	0.0003	0.6343	0.0109	0.5005	0.0083

The best values are shown in bold (Hv = Hypervolume, Th = Threshold).

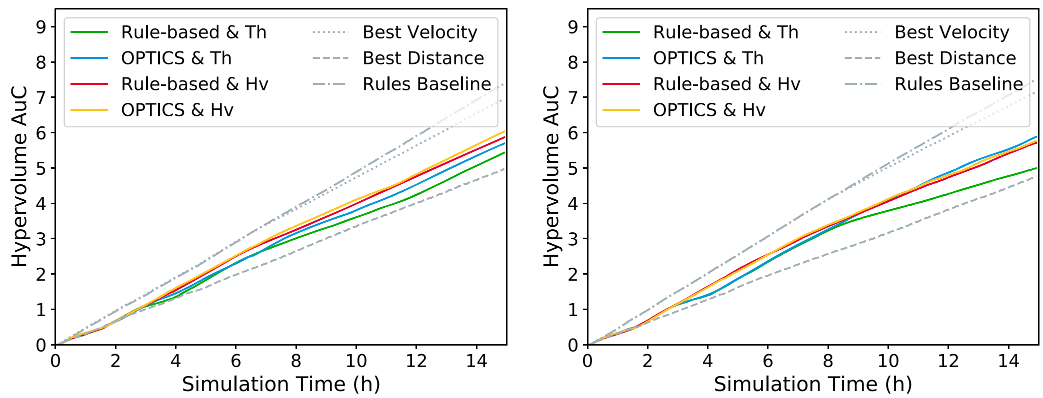
Table 4. Evaluation Summary of the Average and Standard Deviation for Performance Metrics Throughput, Time Loss, Platoon Utilization, and Platoon Time for the Saturday Scenario

Configuration	Throughput		Time Loss		Platoon Utilization		Platoon Time	
	mean	std	mean	std	mean	std	mean	std
Best Distance	0.9945	0.0	0.9255	0.0	0.5999	0.0	0.4522	0.0
Best Velocity	**0.9951**	0.0	**0.9411**	0.0	0.6942	0.0	0.5833	0.0
Fallback Rules	0.9950	0.0	0.9401	0.0	**0.7101**	0.0	**0.6199**	0.0
OPTICS & Hv	0.9949	0.0001	0.9309	0.0004	0.6360	0.0019	0.4918	0.0022
Rule-based & Hv	**0.9950**	0.0001	0.9297	0.0013	0.6367	0.0087	0.4880	0.0137
OPTICS & Th	**0.9950**	0.0000	0.9323	0.0012	**0.6511**	0.0065	**0.5169**	0.0159
Rule-based & Th	**0.9950**	0.0001	**0.9333**	0.0024	0.5677	0.0504	0.4182	0.0520

The best values are shown in bold (Hv = Hypervolume, Th = Threshold).

from 0.4908 to 0.6518 and from 0.4182 to 0.6199 for Wednesday and Saturday, respectively. Again, the fallback rules baseline performs best for both scenarios, and the OPTICS situation detection with Hypervolume and individual thresholds performs best among the framework configurations. The combination of the close average values for all metrics and the small standard deviations does not suggest significant advantages for some configurations. However, this indicates that the framework performs comparably well when considering the results of the baseline, which was designed and configured with complete prior knowledge based on the preliminary situation-dependency study we published [40].

In addition to evaluating individual platooning metrics, we also analyze the progression of the performance over simulation time. Therefore, Figure 13 presents the mean Hypervolume area under curve over simulation time for all configurations and baseline strategies for Wednesday (Figure 13(a)) and Saturday (Figure 13(b)). The baseline strategies are depicted as gray lines with a dotted line for the Best-Velocity, a dashed line for Best-Distance, and a dashed and dotted line for the rules baseline. The colors represent the different configurations. Both plots show a similar result: The Best-Velocity and rules baseline perform best, with a stable increasing gradient of the area under curve, while the Best-Distance baseline performs worst. The curves of the framework configurations do not increase at a constant rate but show more fluctuations in the gradient. All lines are close to each other, but more noticeable differences appear as the simulation progresses. The OPTICS and rule-based situation detection combined with the Hypervolume trigger, perform best for Wednesday. For the Saturday scenario, both configurations perform well again, but OPTICS in combination with individual thresholds outperforms them slightly from 10 hours of simulation time. For both scenarios, the rule-based situation detection in combination with individual thresholds performs worst of all configurations.



(a) Wednesday scenario.

(b) Saturday scenario.

Fig. 13. Mean area under curve evaluation over time for the Hypervolume score of all tested configurations and the baselines on both scenarios. The different colors represent the tested configurations, the x-axis shows the simulation time, and the area under curve is depicted on the y-axis.

The fact that the Best-Velocity and the rules baseline perform best is in line with our case study [40]. This can be explained due to our extensive examination of existing baseline strategies, their configuration, and their performance in various situations and their combination as gold standard strategy. Using this information, we then defined the baseline strategies to represent the best possible performance when complete knowledge of situations, strategies, and configuration was available at design time. However, such intensive studies are not feasible, especially in such dynamic, adaptive use cases. Moreover, it is in the nature of the framework to perform worse than the gold standard, since it needs some time to explore possible strategies and configurations before it can learn and profit from earlier decisions. The better performance of all framework configurations compared to the Best-Distance baseline shows that the framework is able to identify and select a strategy that works well. This reduces the need of expert knowledge or extensive case studies for a use case and, hence, provides a valuable contribution to self-aware optimization.

10.6 Discussion of Further Use Cases

In this section, we want to highlight the generic applicability of the proposed framework by showcasing further use cases for which the framework might be beneficial. The first two use cases can be considered as CPS use cases in the transport and logistics domain, while the third use case originates from the cloud computing research area.

The first use case we want to discuss is the **vehicle routing problem (VRP)**. The classical VRP specifies the assignment of customer orders to vehicles and the optimization of their tours [20], which refers to solving the underlying **Traveling Salesman Problem (TSP)**. Hence, the use case for the framework would be the customer orders, vehicles, and tours. Any optimization algorithm to solve the VRP can be referred to as adaptation planning strategy. The framework would then learn from observed metrics such as the number of orders, the geographical distribution of customers and others, which optimization algorithm, that is, the adaptation planning strategy, would fit best for the current situation.

The second use case is located in the logistics area and covers the optimal planning of warehouses. Working within a mezzanine warehouse consists of two main tasks: (i) filling the storage with goods (storage assignment) and (ii) picking items out of the storage (order picking) [39].

Using the terminology of this article, the warehouse, goods, pickers, and orders are entities in the use case. Any optimization algorithm to plan the storage of goods or the order picking can be considered as adaptation planning algorithms. The framework would receive observation metrics such as the number of goods to be stored, the fill rate of the warehouse, the number of pickers, and others. Then, the framework would learn over time which optimization algorithm, that is, which adaptation planning strategy, fits best for the current situation of the warehouse.

Using the last use case, we want to move on from the logistics and transport domain to a completely novel domain that is cloud computing. This particularly highlights the broad applicability of the proposed framework as a concept. The use case from the cloud computing domain we want to discuss is auto-scaling. The idea is "to have a system that automatically adjusts the resources to the workload handled by the application" [44]. In the terminology of the framework, the resources that would be adjusted could be virtual machines. The auto-scaler, that is, the adaptation planning strategy, analyzes the application and decides when and how many resources to adjust. The framework would receive observation metrics, such as the number of running resources, the number of requests to the application, and others and learn which adaptation planning strategy, that is, which auto-scaler, fits best for the current situation of the application.

10.7 Threats to Validity

In the course of our article, we proposed a set of assumptions that must be met for the framework to be applicable. We already discussed these assumptions in Section 4.1. In addition, we now present and discuss limitations as well as threats to the validity of our evaluation.

First, our framework is intended for application in a broad variety of use cases and therefore provides a use-case-specific adapter to apply it to other examples. However, we limit our evaluation to platooning as representative use case from the ITS domain and did not show results from other use cases. Still, we are convinced that as long as all stated assumptions are met, the framework can also be applied in other use cases and domains due to the provided use case adapter. Therefore, we discuss three additional use cases for which the application of the framework seems to be useful in Section 10.6. Second, we currently only provide a basic algorithm for the strategy selection as well as a limited set of clustering techniques and optimization approaches. We decided to implement these algorithms and approaches, as the selected clustering techniques are commonly used in such scenarios, and the Bayesian optimization performed best in our previous publication [40]. This selection allows us to showcase the potential. However, we do not limit the frameworks functionality to them but rather designed the framework to be modular and would like to encourage future users to extend the framework or individual components and algorithms. Third, we only used one parameter setting for the framework to assess the functionality and performance. Again, we derived this configuration based on our extensive previous case study in platooning and are convinced that this is a good example configuration. Still, we do not claim that we defined the perfect configuration and further evaluation runs can help analyze the validity of the configuration or to find better configurations. Fourth, we limited the time horizon of the scenarios to the first 14 hours of a day and used only one road segment as example. We decided to use the first 14 hours of a day to trade off a long computation time with a minimum set of different traffic situations covered. The selected time horizon includes a low traffic volume at night, a traffic increase until the first rush hour, the decrease to a daytime medium traffic flow, and a final increase towards the second rush hour. Hence, we believe that this time horizon provides sufficiently diverse traffic situations to analyze the functionality of all components. We chose the road segment in Germany, because it was already used in our previous study and we could thus directly transfer the results and gold standards. Evaluations on other road segments can be performed additionally at any time to show the validity of the results. Finally, we limit our evaluation on analyzing the feasibility of the proposed framework. We explicitly exclude an in-depth performance analysis of the framework and its components for this article. Nevertheless, a performance analysis against state-of-the-art approaches is an important evaluation we plan as next step for the future.

We acknowledge that all of the aforementioned threats might limit the transferability of our evaluation results to other use cases. However, we are convinced that we were able to showcase the functionality and usefulness of the proposed framework and can conclude that it has the potential to optimize adaptation planning systems.

11 CONCLUSION

In today's world, circumstances, processes, and requirements for software systems are becoming increasingly complex. To operate properly in such dynamic environments, software systems must adapt to these changes, which has led to the research area of **Self-Adaptive Systems (SAS)**. Platooning is one example of adaptive systems in Intelligent Transportation Systems, which is the ability of vehicles to travel with close inter-vehicle distances. This technology leads to an increase in road throughput and safety, which directly addresses the increased infrastructure needs due to increased traffic on the roads. However, the No-Free-Lunch theorem states that the performance of one platooning coordination strategy is not necessarily transferable to other problems. Moreover, especially in the field of SAS, the selection of the most appropriate strategy depends on the current situation of the system. In this article, we address the problem of self-aware optimization of adaptation planning strategies by designing a framework that includes situation detection, strategy selection, and parameter optimization of the selected strategies. We apply rules and clustering techniques to identify the current situation, as well as Bayesian Optimization to tune the selected strategy's parameters. Further, we learn models of the system and its environment and reason on future decisions based on these models. Finally, we apply the proposed framework on the platooning coordination case study and evaluate the performance of all components of the framework as well as the overall performance of the whole framework.

In the future, we plan to further enhance the components of the framework: First, the coordination component processes the observations from the use case and triggers the other components. However, with increasing runtime of the framework, the amount of data collected from the use case increases. This leads to large datasets that do not necessarily contribute to good performance of the overall system, as the information may become outdated [45, 58]. Hence, it is useful to develop a strategy on how to discard or aggregate the increasing amount of data. Further, the situation detection currently comprises a rule-based and a clustering approach, but is not able to adapt the rule set with learned insights. Hence, a rule-learning mechanism could be applied to improve the rule base of the situation detection. Currently, the strategy selection learns which strategy to choose based solely on all observations on the current situation. However, a global mechanism could provide benefits to the component by adjusting the order of strategies based on the performance of strategies previously experienced in all situations. This could reduce the trial-and-error phase for new situations and, thus, shorten the time to convergence. The parameter optimization component currently provides the hypervolume metric and individual thresholds. However, for other use cases, other techniques for multi-objective optimization could be useful, such as the concept of Pareto-optimality to provide the operator with a set of equally well-performing configurations. Further, approaches to reduce the search space for parameter tuning such as References [24, 50] could speed up the component. In general, we could apply forecasting techniques [68] to anticipate future developments of the system and its environments to proactively plan adaptations. In summary, we developed the framework using components, which allows for dynamic evolution of each component according to the individual requirements and best practices of the targeted use case.

REFERENCES

[1] bast (Bundesanstalt für Straßenwesen) - Automatische Zählstellen 2018. Retrieved from https://www.bast.de/BASt_2017/DE/Verkehrstechnik/Fachthemen/v2-verkehrszaehlung/Daten/2018_1/Jawe2018.html.

[2] PTVPlanung Transport Verkehr AG. Regionaler Nahverkehrsplan Mittleres Mecklenburg/Rostock. Retrieved from https://www.planungsverband-rostock.de/wp-content/uploads/2018/07/NVP%5F%5Fbersicht.pdf.

[3] Anant Agarwal, Jason Miller, Jonathan Eastep, David Wentziaff, and Harshad Kasture. 2009. *Self-aware Computing*. Technical Report. Massachusetts Institute of Technology.

[4] Assad Alam. 2011. *Fuel-efficient Distributed Control for Heavy Duty Vehicle Platooning*. Ph.D. Dissertation. KTH Royal Institute of Technology, Stockholm.

[5] Salem Alelyani, Jiliang Tang, and Huan Liu. 2014. Feature selection for clustering: A review. *Data Cluster.: Algor. Applic.* 29 (2014), 110–121.

[6] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. 2016. ASlib: A benchmark library for algorithm selection. *Artif. Intell.* 237 (2016), 41–58. DOI: https://doi.org/10.1016/j.artint.2016.04.003

[7] Jörg Breithut. A8 zwischen Stuttgart und Leonberg: Polizei stellt Autobahn-Blitzer wieder auf. Retrieved from https://www.stuttgarter-nachrichten.de/inhalt.a8-zwischen-stuttgart-und-leonberg-polizei-stellt-autobahn-blitzer-wieder-auf.631561fb-8f7f-4881-a4cc-74eac1f4a158.html.

[8] Radu Calinescu, Raffaela Mirandola, Diego Perez-Palacin, and Danny Weyns. 2020. Understanding uncertainty in self-adaptive systems. In *Proceedings of the IEEE International Conference on Autonomic Computing and Self-organizing Systems*. IEEE, 242–251. DOI: https://doi.org/10.1109/ACSOS49614.2020.00047

[9] Jinlong Chai, Jiangeng Chang, Yakun Zhao, and Honggang Liu. 2019. An auto-ML framework based on GBDT for lifelong learning. *arXiv preprint arXiv:1908.11033* (2019).

[10] Shelvin Chand, Quang Huynh, Hemant Singh, Tapabrata Ray, and Markus Wagner. 2018. On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems. *Inf. Sci.* 432 (2018), 146–163. DOI: https://doi.org/10.1016/j.ins.2017.12.013

[11] Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee. 2009. *Software Engineering for Self-adaptive Systems: A Research Roadmap*. Springer Berlin.

[12] Radu Chis, Maria Vintan, and Lucian Vintan. 2013. Multi-objective DSE algorithms' evaluations on processor optimization. In *Proceedings of the 9th International Conference on Intelligent Computer Communication and Processing*. IEEE, 27–33. DOI: https://doi.org/10.1109/ICCP.2013.6646076

[13] Michael T. Cox. 2005. Metacognition in computation: A selected research review. *Artif. Intell.* 169, 2 (2005), 104–141. DOI: https://doi.org/10.1016/j.artint.2005.10.009

[14] dpa/lsw. Verkehr - Stuttgart - Meistbefahrener Autobahnabschnitt: Unfallzahlen verdoppelt - Wirtschaft - SZ.de. Retrieved from https://www.sueddeutsche.de/wirtschaft/verkehr-stuttgart-meistbefahrener-autobahnabschnitt-unfallzahlen-verdoppelt-dpa.urn-newsml-dpa-com-20090101-170806-99-537657.

[15] Mica R. Endsley. 2017. Toward a theory of situation awareness in dynamic systems. *Hum. Factors: J. Hum. Factors Ergon. Societ.* 37, 1 (2017), 32–64. DOI: https://doi.org/10.1518/001872095779049543

[16] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2015. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

[17] Erik M. Fredericks, Ilias Gerostathopoulos, Christian Krupitzer, and Thomas Vogel. 2019. Planning as optimization: Dynamically discovering optimal configurations for runtime situations. In *Proceedings of the 13th International Conference on Self-adaptive and Self-organizing Systems*. IEEE, 1–10. DOI: https://doi.org/10.1109/SASO.2019.00010

[18] Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, Adam Hujecek, Frantisek Plasil, and Dominik Skoda. 2017. Strengthening adaptation in cyber-physical systems via meta-adaptation strategies. *ACM Trans. Cyber-Phys. Syst.* 1, 3 (2017), 1–25. DOI: https://doi.org/10.1145/2823345

[19] Adam Ghandar, Zbigniew Michalewicz, Martin Schmidt, Thuy-Duong To, and Ralf Zurbrugg. 2009. Computational intelligence for evolving trading rules. *IEEE Trans. Evolut. Computat.* 13, 1 (2009), 71–86. DOI: https://doi.org/10.1109/TEVC.2008.915992

[20] Bruce L. Golden, Subramanian Raghavan, Edward A. Wasil, et al. 2008. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Vol. 43. Springer.

[21] Mathieu Guillame-Bert, Sebastian Bruch, Josh Gordon, and Jan Pfeifer. 2021. Introducing TensorFlow Decision Forests. Retrieved from https://blog.tensorflow.org/2021/05/introducing-tensorflow-decision-forests.html.

[22] Tobias Hardes and Christoph Sommer. 2019. Dynamic platoon formation at urban intersections. In *Proceedings of the 44th IEEE Conference on Local Computer Networks*. 101–104. DOI: https://doi.org/10.1109/LCN44214.2019.8990846

[23] Tobias Hardes and Christoph Sommer. 2019. Towards heterogeneous communication strategies for urban platooning at intersections. In *Proceedings of the IEEE Vehicular Networking Conference*. IEEE, 1–8. DOI: https://doi.org/10.1109/VNC48660.2019.9062835

[24] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, Carlos A. Coello Coello (Ed.). Springer Berlin, 507–523. DOI: https://doi.org/10.1007/978-3-642-25566-3_40

[25] Sehyeok Kang, Taeyeong Choi, and Theodore P. Pavlic. 2020. How far should I watch? Quantifying the effect of various observational capabilities on long-range situational awareness in multi-robot teams. In *Proceedings of the IEEE International Conference on Autonomic Computing and Self-organizing Systems*. IEEE, 146–152. DOI: https://doi.org/10.1109/ACSOS49614.2020.00036

[26] Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. 2019. Automated algorithm selection: Survey and perspectives. *Evolut. Computat.* 27, 1 (2019), 3–45. DOI: https://doi.org/10.1162/evco_a_00242

[27] Pascal Kerschke and Heike Trautmann. 2019. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolut. Computat.* 27, 1 (03 2019), 99–127. DOI: https://doi.org/10.1162/evco_a_00236

[28] Cody Kinneer, Zack Coker, Jiacheng Wang, David Garlan, and Claire Le Goues. 2018. 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*. 40–50. DOI: https://doi.org/10.1145/3194133.3194145

[29] Lars Kotthoff, Pascal Kerschke, Holger Hoos, and Heike Trautmann. 2015. Improving the state of the art in inexact TSP solving using per-instance algorithm selection. In *Learning and Intelligent Optimization*, Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eléonore Marmion (Eds.). Springer International Publishing, Cham, 202–217. DOI: https://doi.org/10.1007/978-3-319-19084-6_18

[30] Samuel Kounev, Peter Lewis, Kirstie L. Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, et al. 2017. The notion of self-aware computing. In *Self-aware Computing Systems*. Springer, 3–16.

[31] Jeff Kramer and Jeff Magee. 2007. Self-managed systems: An architectural challenge. In *Proceedings of the Future of Software Engineering Conference*. IEEE, 259–268. DOI: https://doi.org/10.1109/FOSE.2007.19

[32] Christian Krupitzer, Veronika Lesch, Martin Pfannemüller, Christian Becker, and Michele Segata. 2019. A modular simulation framework for analyzing platooning coordination. In *Proceedings of the 1st ACM Workshop on Technologies, mOdels, and Protocols for Cooperative Connected Cars (TOP-Cars), Colocated with ACM MobiHoc*. ACM.

[33] Christian Krupitzer, Felix Maximilian Roth, Sebastian Vansyckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervas. Mob. Comput.* 17 (2015), 184–206. DOI: https://doi.org/10.1016/j.pmcj.2014.09.009

[34] Christian Krupitzer, Michele Segata, Martin Breitbach, Samy El-Tawab, Sven Tomforde, and Christian Becker. 2018. Towards infrastructure-aided self-organized hybrid platooning. In *Proceedings of the IEEE Global Conference on AI & IoT*.

[35] Veronika Lesch. 2020. Toward a framework for self-learning adaptation planning through optimization. In *Organic Computing: Doctoral Dissertation Colloquium 2020*. Kassel University Press GmbH, 17–31.

[36] Veronika Lesch, Martin Breitbach, Michele Segata, Christian Becker, Samuel Kounev, and Christian Krupitzer. 2021. An overview on approaches for coordination of platoons. *IEEE Trans. Intell. Transport. Syst.* (2021). Early Access on IEEE Xplore.

[37] Veronika Lesch, Marius Hadry, Samuel Kounev, and Christian Krupitzer. 2021. *A Case Study on Optimization of Platooning Coordination*. Technical Report. Universität Würzburg and Universität Hohenheim. Q13

[38] Veronika Lesch, Christian Krupitzer, Kevin Stubenrauch, Nico Keil, Christian Becker, Samuel Kounev, and Michele Segata. 2021. A comparison of mechanisms for compensating negative impacts of system integration. *Fut. Gen. Comput. Syst.* 116 (Mar. 2021), 117–131.

[39] Veronika Lesch, Patrick B. M. Müller, Moritz Krämer, Samuel Kounev, and Christian Krupitzer. 2021. *A Case Study on Optimization of Warehouses*. Technical Report.

[40] Veronika Lesch, Tanja Noack, Johannes Hefter, Samuel Kounev, and Christian Krupitzer. 2021. Towards situation-aware meta-optimization of adaptation planning strategies. In *Proceedings of the 2nd IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS'21)*. IEEE. Q14

[41] Peter Lewis, Kirstie L. Bellman, Christopher Landauer, Lukas Esterle, Kyrre Glette, Ada Diaconescu, and Holger Giese. 2017. Towards a framework for the levels and aspects of self-aware computing systems. In *Self-aware Computing Systems*. Springer, 51–85. DOI: https://doi.org/10.1007/978-3-319-47474-8_3

[42] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.* 18, 1 (2017), 6765–6816.

[43] Wei Liu, Seong-Woo Kim, Scott Pendleton, and Marcelo H. Ang. 2015. Situation-aware decision making for autonomous driving on urban road using online POMDP. In *Proceedings of the IEEE Intelligent Vehicles Symposium*. IEEE, 1126–1133. DOI: https://doi.org/10.1109/IVS.2015.7225835

[44] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A. Lozano. 2014. A review of auto-scaling techniques for elastic applications in cloud environments. *J. Grid Comput.* 12, 4 (2014), 559–592.

[45] Shaul Markovitch and Paul D. Scott. 1988. The role of forgetting in learning. In *Machine Learning Proceedings 1988*, John Laird (Ed.). Morgan Kaufmann, San Francisco, CA, 459–465. DOI: https://doi.org/10.1016/B978-0-934613-64-4.50052-9

[46] Christoph Neumüller, Andreas Scheibenpflug, Stefan Wagner, Andreas Beham, and Michael Affenzeller. 2012. Large scale parameter meta-optimization of metaheuristic optimization algorithms with heuristiclab Hive. *Actas Del VIII Español Sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados* (2012).

[47] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. 2012. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Trans. Evolut. Computat.* 17, 5 (2012), 621–639. DOI: https://doi.org/10.1109/TEVC.2012.2227326

[48] Gilles Perrouin, Brice Morin, Franck Chauvel, Franck Fleurey, Jacques Klein, Yves Le Traon, Olivier Barais, and Jean-Marc Jézéquel. 2012. Towards flexible evolution of dynamically adaptive systems. In *Proceedings of the 34th International Conference on Software Engineering*. IEEE, 1353–1356. DOI: https://doi.org/10.1109/ICSE.2012.6227081

[49] Barry Porter and Roberto Rodrigues Filho. 2016. Losing control: The case for emergent software systems using autonomous assembly, perception, and learning. In *Proceedings of the 10th International Conference on Self-adaptive and Self-organizing Systems*. IEEE, 40–49. DOI: https://doi.org/10.1109/SASO.2016.10

[50] Dmytro Pukhkaiev and Sebastian Götz. 2018. BRISE: Energy-efficient benchmark reduction. In *Proceedings of the 6th International Workshop on Green and Sustainable Software*. 23–30. DOI: https://doi.org/10.1145/3194078.3194082

[51] John R. Rice. 1976. The algorithm selection problem. In *Advances in Computers*. Vol. 15. Elsevier, 65–118. DOI: https://doi.org/10.1016/S0065-2458(08)60520-3

[52] Tom Robinson, Eric Chan, and Erik Coelingh. 2010. Operating platoons on public motorways: An introduction to the SARTRE platooning programme. In *Proceedings of the 17th World Congress on Intelligent Transport Systems*.

[53] Matthias Rockl, Patrick Robertson, Korbinian Frank, and Thomas Strang. 2007. An architecture for situation-aware driver assistance systems. In *Proceedings of the IEEE 65th Vehicular Technology Conference*. IEEE, 2555–2559. DOI: https://doi.org/10.1109/VETECS.2007.526

[54] Michele Segata, Stefan Joerer, Bastian Bloessl, Christoph Sommer, Falko Dressler, and Renato Lo Cigno. 2014. PLEXE: A platooning extension for veins. In *Proceedings of the IEEE Vehicular Networking Conference*. 53–60.

[55] Kate A. Smith-Miles. 2009. Cross-disciplinary perspectives on meta-learning for algorithm selection. *Comput. Surv.* 41, 1 (2009), 1–25. DOI: https://doi.org/10.1145/1456650.1456656

[56] Christoph Sommer, Reinhard German, and Falko Dressler. 2011. Bidirectionally coupled network and road traffic simulation for improved IVC analysis. *IEEE Trans. Mob. Comput.* 10, 1 (2011), 3–15.

[57] Timo Sturm, Christian Krupitzer, Michele Segata, and Christian Becker. 2021. A taxonomy of optimization factors for platooning. *IEEE Trans. Intell. Transport. Syst.* 22, 10 (2021), 6097–6114. DOI: https://doi.org/10.1109/TITS.2020.2994537

[58] Sergey Sukhov, Mikhail Leontev, Alexander Miheev, and Kirill Sviatov. 2020. Prevention of catastrophic interference and imposing active forgetting with generative methods. *Neurocomputing* 400 (2020), 73–85. DOI: https://doi.org/10.1016/j.neucom.2020.03.024

[59] Xudong Sun, Jiali Lin, and Bernd Bischl. 2019. ReinBo: Machine learning pipeline search and configuration with Bayesian optimization embedded reinforcement learning. *arXiv preprint arXiv:1904.05381* (2019).

[60] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. 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*. 847–855. DOI: https://doi.org/10.1145/2487575.2487629

[61] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *J. Roy. Statist. Societ.: Series B (Statist. Methodol.)* 63, 2 (2001), 411–423. DOI: https://doi.org/10.1111/1467-9868.00293

[62] Lucian Vinţan, Radu Chiş, Muhammad Ali Ismail, and Cristian Coţofană. 2015. Improving computing systems automatic multiobjective optimization through meta-optimization. *IEEE Trans. Comput.-aid. Des. Integ. Circ. Syst.* 35, 7 (2015), 1125–1129. DOI: https://doi.org/10.1109/TCAD.2015.2501299

[63] Shuai Wang, Shaukat Ali, Tao Yue, Yan Li, and Marius Liaaen. 2016. A practical guide to select quality indicators for assessing Pareto-based search algorithms in search-based software engineering. In *Proceedings of the 38th International Conference on Software Engineering*. 631–642.

[64] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. Göschka. 2013. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-adaptive Systems II*. Springer, 76–107.

[65] David H. Wolpert and William G. Macready. 1997. No free lunch theorems for optimization. *IEEE Trans. Evolut. Computat.* 1, 1 (1997), 67–82. DOI: https://doi.org/10.1109/4235.585893

[66] Xiao-Feng Xie, Stephen F. Smith, Gregory J. Barlow, and Ting-Wei Chen. 2014. Coping with real-world challenges in real-time urban traffic control. In *Proceedings of the 93rd Annual Meeting of the Transportation Research Board*. 1–15.

[67] Yuanyuan Zhang, Mark Harman, Gabriela Ochoa, Guenther Ruhe, and Sjaak Brinkkemper. 2018. An empirical study of meta- and hyper-heuristic search for multi-objective release planning. *ACM Trans. Softw. Eng. Methodol.* 27, 3 (2018). DOI: https://doi.org/10.1145/3196831

[68] Marwin Züfle, André Bauer, Veronika Lesch, Christian Krupitzer, Nikolas Herbst, Samuel Kounev, and Valentin Curtef. 2019. Autonomic forecasting method selection: Examination and ways ahead. In *Proceedings of the 16th IEEE International Conference on Autonomic Computing*. IEEE.

Received 14 March 2022; revised 17 September 2022; accepted 28 September 2022