A Literature Review on Optimization Techniques for Adaptation Planning in Adaptive Systems: State of the Art and Research Directions

Elia Henrichs[a], Veronika Lesch[b], Martin Straesser[b], Samuel Kounev[b], Christian Krupitzer[a,*]

[a]*Department of Food Informatics & Computational Science Lab, University of Hohenheim, Stuttgart, Germany*
[b]*Software Engineering Group, University of Würzburg, Würzburg, Germany*

Abstract

Context: Recent developments in modern IT systems including internet of things, edge/fog computing, or cyber-physical systems support intelligent and seamless interaction between users and systems. This requires a reaction to changes in their environment or the system. Adaptive systems provide mechanisms for these reactions.

Objective: To implement this functionality, several approaches for the planning of adaptations exist that rely on rules, utility functions, or advanced techniques, such as machine learning. As the adaptation space with possible options is often extensively huge, optimization techniques might support efficient determination of the adaptation space and identify the system's optimal configuration. With this paper, we provide a systematic review of adaptation planning as the optimization target.

Method: In this paper, we review which optimization techniques are applied for adaptation planning in adaptive systems using a systematic literature review approach.

Results: We reviewed 115 paper in detail out of an initial search set of 9,588 papers. Our analysis reveals that learning techniques and genetic algorithms are by far dominant; in total, heuristics (anytime learning) are more frequently applied as exact algorithms. We observed that around 57% of the approaches target multi-objectiveness and around 30% integrate distributed optimization. As last dimension, we focused on situation-awareness, which is only supported by two approaches.

Conclusion: In this paper, we provide an overview of the current state of the art of approaches that rely on optimization techniques for planning adaptations in adaptive systems and further derive open research challenges, in particular regarding the integration of distributed optimization and situation-awareness.

Keywords: Self-adaptive Systems, Adaptation Planning, Optimization, Survey

1. Introduction

Cyber-physical systems, industrial internet/industry 4.0, internet of things, smart city, smart grid, and self-driving vehicles are only a few examples showing that the world is transitioning towards integrating adaptive systems into our everyday life. Those systems inherently require interacting with the environment and reacting autonomously to changes in the environment for fulfilling service-level agreements or maintaining a sufficient Quality of Service level. Therefore, those systems must be adaptive.

Self-adaptive (software) systems are able to change their behavior at runtime as a response to changes in their environment or in the system itself [1, 2]. Those systems can work in dynamic and uncertain environments. They are often divided into a managed subsystem, i.e., software and hardware resources that interact with the users or back-end systems, and a managing subsystem, which is able to control and adapt the managed subsystem. As a de facto standard, the managing subsystem implements the Monitor-Analyze-Plan-Execution (MAPE) system model [3] for

*Corresponding author: phone: +49-711-459-23664; fax: +49-711-459-24433;
Email addresses: elia.henrichs@uni-hohenheim.de (Elia Henrichs), veronika.lesch@uni-wuerzburg.de (Veronika Lesch), martin.straesser@uni-wuerzburg.de (Martin Straesser), samuel.kounev@uni-wuerzburg.de (Samuel Kounev), christian.krupitzer@uni-hohenheim.de (Christian Krupitzer)

structuring the management functionality into (i) monitoring the environment and the system resources, (ii) analyzing if an adaptation is required, (iii) planning the necessary adaptation actions, and (iv) executing those actions.

However, the planning process can be complex due to several circumstances. First, a large configuration space of the parameters of self-adaptive systems [4] complicates describing and testing all possible adaptation options. Second, an unpredictable number of possible environmental situations prohibits testing all adaptation options at design time and necessitates planning at run time to identify an appropriate configuration for a given situation dynamically [5]. Classical, non-iterative machine learning techniques (e.g., for classification or clustering) can only support this insufficiently because they require a large training set. Further, learning on-the-fly might decrease the availability of the systems because this includes that the systems come into a situation where adaptation is required because the learner can only learn from situations. As a result, adaptation planning must rely on simple models that capture a system's adjustable input parameters and possible observations (output parameters). Following this idea, we proposed in [6] *planning as optimization*: the use of optimization strategies to discover optimal system configurations.

This paper aims to provide a systematic overview and analysis of the current state of the art of planning as optimization, i.e., the application of mathematical, statistical, or nature-inspired optimization techniques for adaptation planning. First, this work provides a general overview of the used techniques. Second, the environment of self-adaptive systems might change frequently requiring anytime learning supported techniques [7], i.e., techniques that constantly provide a usable solution. Hence, anytime algorithms which are able to achieve this in case of optimization [8] should be applied. Third, adaptive systems are often composed of several sub-systems, as most adaptive systems at least separate the adaptation logic from the managed elements as this improves maintainability [9]. Additionally, adaptive systems are often highly distributed [10] and have multiple stakeholders with (potentially conflicting) objectives. This might demand integrating distributed optimization techniques in case that the adaptation control cannot be centralized. Lastly, the "*No-Free-Lunch-Theorem*" [11] describes that there is no general optimization approach that performs best in all scenarios; rather the pattern of data highly influences the choice of the optimization technique. Based on this theorem, we showed in our previous works [6, 12] the necessity to provide a situation-aware change of the used adaptation techniques, as different situations might have other characteristic data patterns. Accordingly, we discuss the current integration of situation-awareness. In summary, this paper contributes to the body of research by addressing:

- Overview of the application of optimization techniques for adaptation planning in adaptive systems.
- Analysis of anytime learning supported applications for adaptation planning in adaptive systems.
- Analysis of multi-objectiveness supported applications for adaptation planning in adaptive systems.
- Analysis of distributed optimization supported applications for adaptation planning in adaptive systems.
- Discussion of situation-awareness support for adaptation planning in adaptive systems.

The remainder of this paper is structure as follows. Section 2 explains several fundamentals related to optimization in general and the specifics of self-adaptive systems. Section 3 presents the methodological approach for the literature review. Subsequently, Section 4 provides an overview of the identified optimization techniques for answering research question RQ1. In Section 5, we discuss the results of the literature review with respect to anytime learning [7], multi-objectiveness, distributed optimization, and situation-awareness for answering the research questions RQ2, RQ3, RQ4, and RQ5. Further, Section 6 discusses threats to validity. We distinguish our work against other surveys in the field in Section 7. Finally, Section 8 concludes this paper with a summary of our results.

2. Fundamentals

In the following section, we provide a short overview of the terminology and selected methods for solving optimization problems relevant for the remainder of this work. Furthermore, we provide a definition of self-adaptive (software) systems as the targeted system domain for this study.

2.1. General Terminology of Optimization Problems

In an optimization problem, we aim to find an optimal system state x^* using a set of influenceable variables [13]. These variables $x_1, x_2, ..., x_n$ are composed into the *design vector* X. The set of all possible values of X is called *design space* D, while the subset of D, where we are looking for an optimal configuration, is called *search space*. We

determine the quality of a state X using an *objective function* $f(X) : D \rightarrow \mathbb{R}$. Depending on the actual formulation of the problem, a state x^* is globally (locally) optimal if $f(x^*)$ is a global (local) maximum or minimum of f. Note that if x^* is a maximum of f, then x^* is a minimum of $-f$ so that a maximization problem is transformed into a minimization problem. For practical reasons, a minimum is usually targeted. We can state an optimization problem as:

$$\text{Find} \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \text{which minimizes} \quad f(X) \tag{1}$$

The problem denoted in (1) is called an *unconstrained optimization problem*. In contrast, a *constrained optimization problem* is present if the design variables have to fulfill several constraints $g_1(X), g_2(X), ..., g_m(X)$. This is often the case in practical-oriented applications. Additionally, an optimization problem with more than one objective function or an objective function which combines several objectives is called a *multi-objective optimization problem*.

2.2. Self-Adaptive Systems and Related Concepts

In this paper, we mainly target the domain of self-adaptive (software) systems. However, several closely related concepts provide similar approaches. Hence, we also included those in the literature review. In the following, we present the basics of those different research streams.

2.2.1. Self-Adaptive Systems

The literature provides several definitions for the term *self-adaptive (software) systems* [1, 14, 15, 16, 17] as well as different terms that are used interchangeably: Dynamically Adaptive Systems (e.g., [18]), Autonomic Systems (e.g., [16]), Self-managing Systems (e.g., [16]), Self-adaptive Systems (e.g., [1]), or Self-adaptive Software Systems (e.g., [15]). This paper consistently uses the term *self-adaptive system* and follows the definition of the first Dagstuhl Seminar on Software Engineering for Self-Adaptive Systems:

> *[Self-adaptive systems] are able to adjust their behaviour in response to their perception of the environment and the system itself. The "self" prefix indicates that the systems decide autonomously (i.e., without or with minimal interference) how to adapt or organize to accommodate changes in their contexts and environments.* ([1, p. 1];[17, p. 49])

From an architectural point of view (see Figure 1), a self-adaptive system is composed of two parts [1, 19]: a managing system – the adaptation logic – that controls the second part, the managed resources, a set of software and hardware resources, e.g., servers, laptops, smartphones, robots, or unmanned vehicles. Therefore, the adaptation logic observes the environment and the managed resources, analyzes the need for adaptation, plans such adaptations and controls the execution of the adaptation. These four steps (also called MAPE or MAPE control feedback loop) may share and/or exploit the knowledge (becoming MAPE-K) built from the monitored environment, the analyzed information, the planned changes, and the result of the execution of the changes. This knowledge may grow in time, being enriched with new information about the environment and the applied adaptations. The MAPE control feedback loop is the de facto standard for the design of the adaptation logic for self-adaptive systems [16]. Other authors propose similar feedback structures, such as the sense-plan-act control [20], the autonomic control loop [21], or the observer/controller architecture [22]. The authors of [2] discuss properties of self-adaptation that influence the implementation of self-adaptive systems.

2.2.2. Related Research Streams

Other research streams focus on systems with similar properties as self-adaptive systems. The most related concept is Autonomic Computing [3]. Researchers in the Autonomic Computing domain integrate principles from biology, mainly from the autonomous nervous system, to equip systems with autonomic capabilities. The MAPE control loop arose in the Autonomic Computing area.

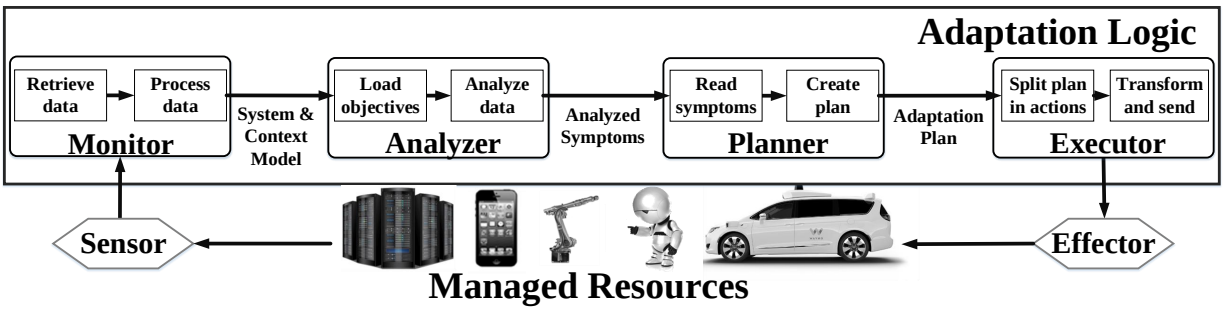

Figure 1: A conceptual model of a self-adaptive system.

The authors of [23] define *self-aware computing systems* as systems that (i) reason on the knowledge of self-awareness and (ii) act accordingly. This makes their definition of self-aware computing systems identical to this paper's view on self-adaptive systems. One has to distinguish this concept from self-awareness as defined in [16]: There, self-awareness – capturing knowledge on itself – is seen as underlying system property for self-adaptation, i.e., acting on self-awareness and context-awareness.

Babaoglu and Shrobe [24] see self-adaptive systems as top-down systems with central control, whereas self-organizing systems are dedicated units that organize themselves bottom-up without a central instance. However, reviewing the current literature of self-adaptive systems shows that self-adaptive systems offer both centralized and decentralized system control [10, 2]. Consequently, we do not distinguish self-organizing systems and self-adaptive systems in this work as a self-organizing system can be composed of self-adaptive systems.

Organic Computing is associated with systems that use bio-inspired concepts to implement organic behavior [25]. Similar to self-adaptive systems, Organic Computing systems try to achieve self-* properties. In contrast, they focus on (i) the integration of principles from nature-inspired computing, (ii) the emergence of systems for shifting design activities to runtime, and (iii) the human-in-the-loop as a first-class entity rather than an element to avoid. Pervasive/Ubiquitous Computing [26] aims at the seamless integration of information technology and everyday devices to support humans by smart information technology. These systems are often context-aware (i.e., they react to changes in their environment; similar to situation-awareness) and adaptive. However, they target solutions in the Internet-of-Things domain rather than generic systems.

Collective Adaptive Systems (CAS) integrate evolutionary self-organization (i.e., controllability of long-term autonomy), driven forces behind evolution (i.e., coping with the complexity of "natural chemistry"), developmental drift (i.e., incorporating artificial sociality), and long-term homeostasis self-identification (i.e., the emergence of self-organization) [27]. As being a similar research stream, we also included CAS as well as the domain of multi-agent systems in our review.

3. Methodology

Our methodology for the survey adapts some methods from the guidelines of Webster and Watson [28] for a structured literature review and Petersen et al. [29] for systematic mapping studies. Our research is based on the steps shown in Figure 2. In the beginning, we framed our aim in the form of research questions and identified relevant venues. We defined exclusion and inclusion criteria and performed keyword-based searches for filtering the articles based on their titles and abstracts. After identifying the set of relevant papers, descriptions and properties of the optimization techniques as well as bibliography data have been extracted. In the following, we describe these steps.

3.1. Definition of Research Questions

The primary aim of this work is to provide an overview of optimization techniques in the context of adaptive systems. According to this goal, we derived our research questions. To get a picture of used optimization techniques, we divided the optimization approaches into categories adapted from [13] within research question RQ1. Further anytime learning techniques [7] (e.g., using anytime algorithms [8]), provide a solution for adaptation planning using

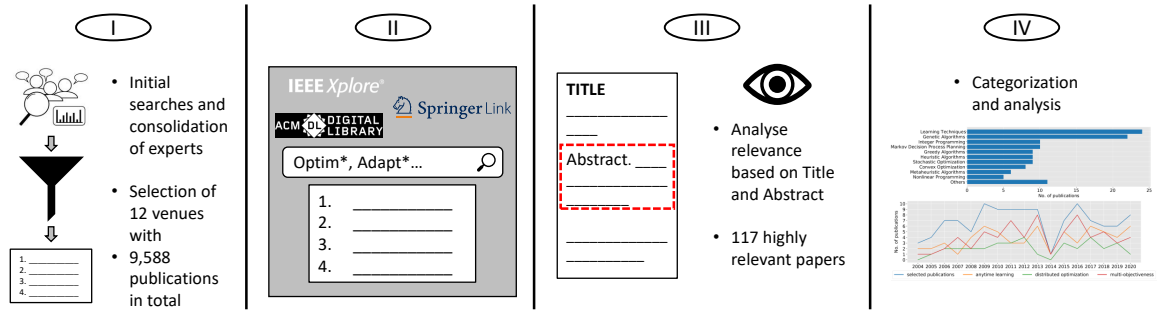

Figure 2: Overview of the methodology for the classification in this survey. An initial search and preliminary selection of venues is followed by an iterative process.

intermediate solutions of the optimizer in contrast to exact algorithms, such as Integer Programming, which only provide a final solution. Those intermediate solutions can fasten the reaction to changes as adaptations can be applied faster compared to exact algorithms which include waiting for the final solution. Because of the dynamics of the environment and system composition of self-adaptive systems, we postulate that anytime learning might be beneficial as research question (RQ2). As self-adaptive systems, especially self-organizing systems such as self-organizing networks or autonomous vehicles that adjust their behavior, might be composed of several different subsystems with many stakeholders, multi-objective scenarios with even potentially conflicting objectives are feasible. Hence, with research question RQ3 we study to what extent multi-objective approaches are present. Additionally, adaptive systems are often distributed [10]. In combination with the mentioned various stakeholders / users, this might demand for the application of distributed optimization techniques in case that the adaptation control cannot be centralized. Accordingly, distributed optimization techniques might support the decision making in distributed adaptive systems. Hence, we formulated research question RQ4. Lastly, the "*No-Free-Lunch-Theorem*" [11] states that there is no general optimization approach that outperforms all other techniques in all possible use cases because the pattern of data determine the performance of an optimization technique. Based on this theorem, we have shown in [6] and [12] that a meta-adaptation of the optimization technique used for adaptation planning might be beneficial as different system situation reflect the required for different optimizers due to the different data patterns in a specific situation. Accordingly, we added this circumstance in research question RQ5. These considerations lead to the following research questions:

RQ1 - Which optimization techniques are applied for adaptation planning in adaptive systems?
RQ2 - To what extent is anytime learning supported?
RQ3 - To what extent is multi-objectiveness supported?
RQ4 - To what extent are distributed optimization techniques integrated?
RQ5 - Does the used techniques for planning as optimization support situation-awareness?

3.2. Selection Method

We investigate how optimization techniques can be used for adaptation planning in self-adaptive systems and related concepts. As the terms "optimization", "optimize", "adaptation planning", and similar terms are often used in the context of self-adaptive system research, we avoided a fully open search term-based literature search as this would result in misleading result (e.g., all papers using self-optimize or self-optimization would be included). Hence, we used a focused, systematic mapping study. In the following, we describe the selection method for building our set of relevant papers. The process is similar to the one used in [30], in which the authors studied the application of learning in collective adaptive systems.

(1) Identification of Relevant Sources. To identify relevant conferences and journals, we performed some initial searches and consolidated experts in the field of study. The selected venues are shown in Table 1. Those include the

conferences AMAAS, ASE, FSE, ICAC, SASO, ACSOS, and SEAMS as well as the journals JAAMAS, IEEE Software, TAAS, TOSEM, and TSE. We focus on reviewing advanced and high-quality studies published in the leading conferences and journals in the self-adaptive systems domains and closely related research communities. Additionally, we included top venues researching autonomous and multi-agent systems (e.g., AAMAS, JAAMAS). However, we do not exhaustively search the entire multi-agent system domain. Accordingly, we used the databases DBLP, IEEExplore, ACM DL, and SpringerLink to access the publications of the venues. We analyzed papers from January 2004, the year in which the *International Conference on Autonomic Computing* (*ICAC*) took place the first time, to December 2020. We only analyzed full research papers or articles for this survey and excluded texts such as editorials, demonstrations, short papers, workshop contributions, letters, or posters because those often report work-in-progress. Table 2 shows the initial number of articles after this filtering process.

Table 1: Selected Venues for the Survey.

Type	Acronym	Full Name
Conference	AAMAS	International Conference of Autonomous Agents and Multi-Agent Systems
	ASE	International Conference on Automated Software Engineering
	FSE	ACM SIGSOFT International Symposium on the Foundations of Software Engineering
	ICAC	IEEE International Conference on Autonomic Computing
	SASO	IEEE International Conference on Self-Adaptive and Self-Organizing Systems
	ACSOS	IEEE International Conference on Autonomic Computing and Self-Organizing Systems
	SEAMS	ICSE Symposium on Software Engineering for Adaptive and Self-Managing Systems
Journal	JAAMAS	Springer Autonomous Agents and Multi-Agent Systems
	Softw.	IEEE Software
	TAAS	ACM Transactions on Autonomous and Adaptive Systems
	TOSEM	ACM Transactions on Software Engineering and Methodology
	TSE	IEEE Transactions on Software Engineering

(2) Definition of Coarse-Grained Exclusion and Inclusion Criteria. First, a set of keywords was defined for an initial screening. This set was continuously updated with further relevant keywords found during this screening. The set contained two groups of keywords: The first group concerned the term "optimization" consisting of the keywords with prefixes *optim-*, *maxim-*, *minim-* and *best-*. The second group providing the relation to adaptive systems is captured with prefixes including *adapt-*, *self-*, *aware*, *transform-* and *autonom-* among others, caused by the variety of terms. However, even if we detected no match with this second group of search targets, we included the paper in our analysis in case we observed a high match with the specified terms. Consequently, we excluded (i) papers without a hit of a keyword, (ii) works focusing on self-optimization or (iii) that did not explicitly address the adaptation planning.

(3) Study Selection Procedure. Based on the mentioned criteria, we performed a rough filtering of the articles considering their titles, abstracts, and, if applicable, linked keywords. For each paper fitting the scope of the inclusion criteria, we conducted a full-text filtering. Two reviewers conducted the paper selection and analyzed the sampled studies to confirm their relevance. In case of doubt, advice from the other co-authors was taken into account.

3.3. Analysis Method

For each remaining article, we extracted the used optimization technique as well as the specific method/algorithm for planning as optimization in the adaptation planning step based on the categories for optimization techniques from [13]. These categories were convex programming as classic optimization technique; nonlinear programming in general and heuristic methods in particular as nonlinear optimization techniques; integer programming optimization techniques; stochastic programming and Markov processes as stochastic optimization techniques; and genetic algorithms and learning techniques as further optimization techniques. This information is used to answer research question RQ1 (see Section 4). Additionally, we extracted the following categories (see Section 5):

- **Anytime Algorithm** (RQ2): Does an approach support anytime learning?

Table 2: Overview of analyzed and included publications per year and venue. Gray cells mean that the venue was not available. Note that in 2020 the gray cells indicate the ACSOS, which is a merge of ICAC and SASO and replaces them.

	Year																	
Venue	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	
AAMAS	273	130	250	126	254	260	163	125	282	140	167	166	146	160	194	193	194	3223
	1	1	-	3	-	-	2	1	2	-	-	4	1	-	-	-	1	16
ASE	25	28	22	36	36	38	34	37	21	73	82	76	71	83	76	91	93	922
	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	-	3
FSE	26	29	25	49	32	35	33	45	46	72	61	97	74	99	104	95	127	1049
	-	-	-	-	-	1	1	1	-	-	-	-	1	-	-	-	-	4
ICAC	31	25	26	15	18	15	23	20	18	35	29	37	23	27	15	16	18	391
	2	1	6	2	1	3	3	1	2	1	-	1	3	1	1	-	3	31
SASO				28	42	27	25	21	26	26	16	14	14	14	15	12		280
				-	-	3	2	1	1	1	-	-	1	1	-	1		11
SEAMS			13	18	17	17	13	26	19	19	18	19	18	21	29	23	26	296
			-	-	1	1	-	-	-	2	1	1	2	2	2	1	2	15
JAAMAS	16	24	25	26	29	31	30	32	32	30	30	36	38	48	25	25	55	532
	-	2	1	1	1	-	-	3	2	1	-	-	1	-	-	1	2	15
Softw.	76	74	74	63	60	69	64	69	66	61	78	68	77	74	78	81	80	1212
	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	2
TAAS			8	14	19	21	14	27	39	14	25	19	26	20	19	16	7	288
			-	-	1	1	1	-	1	2	-	1	1	2	1	2	-	13
TOSEM	10	12	12	15	20	13	13	18	18	35	44	22	16	12	23	23	31	337
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	1
TSE	71	73	58	53	52	51	49	48	80	95	63	67	60	60	58	58	62	1058
	-	-	-	-	1	-	-	1	-	2	-	-	-	-	-	-	-	4
	528	395	513	443	579	577	461	468	647	600	613	621	563	618	636	633	693	9588
	3	4	7	7	5	9	9	9	8	9	1	7	10	7	6	6	8	115

- **Single/Multi-objectiveness** (RQ3): Does an approach support single- or multi-objectiveness?
- **Distributed Optimization** (RQ4): Is the optimization technique distributed on several subsystems, i.e., for distributed adaptive systems?
- **Situation-awareness** (RQ5): Is the approach situation-aware, i.e., is it possible to switch or adjust the optimization algorithm depending on the environment/system situation?

3.4. Selected Studies

As mentioned, we focus on seven conferences and five journals (see Table 1) as those provide high-quality publications in the research field. The publication range spanned works from 2004 to 2020. In total, we analyzed 9,588 publications, from which we included 115 after the application of the inclusion and exclusion criteria. From the selected publications, around 70% were originally published at conferences (see Figure 3). Having only very few articles in the main software engineering journals might be surprising at first. However, we think that this is not too surprising as those journals cover a broad range of topics besides adaptive systems engineering. In contrast, the conference are more focused to either adaptive systems or optimization procedures in general, which can explain the higher amount of relevant publications.

4. Using Optimization in Self-adaptive Systems

Many approaches apply optimization techniques in self-adaptive systems to generate new system configurations or adaptation plans, mainly within the planning procedure. To provide an overview of the applied techniques, we performed an analysis of these techniques based on approaches published since 2004 in highly rated conferences and journals related to the research communities of adaptive systems (see Table 1). The techniques were classified into a

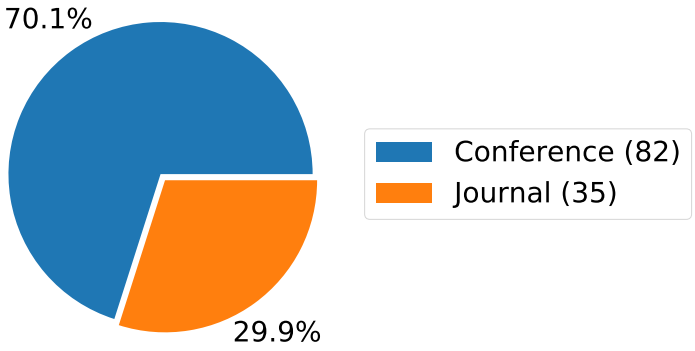

Figure 3: Share of the publication type

set of previously defined categories and the resulting frequencies were compared answering the first research question:

RQ1 - Which optimization techniques are applied for adaptation planning in adaptive systems?

Depending on the types of the objective function, design variables, and constraints, multiple variants of optimization problems emerge. There is no known method which produces solutions for all those variants efficiently. In adaptive systems, we often have to deal with limited computational power and bounded reaction times. Thus, an optimization technique, which suits the application context and requirements, has to be found. As shown in Figure 4, we classified the techniques into 11 categories: Learning Techniques, Genetic Algorithms, Integer Programming, Markov Decision Process Planning, Greedy Algorithms, Heuristic Algorithms, Stochastic Optimization, Convex Optimization, Metaheuristic Algorithms, Nonlinear Programming and Miscellaneous Approaches (Others), which could not be assigned to the categories previously mentioned.

Next, we summarize the findings of the literature review w.r.t. the identified optimization techniques. The classification of optimization techniques is often not disjoint as many approaches use techniques from several categories. We classified them according to their primary technique used. Nevertheless, we assigned some publications in multiple categories because they were using several optimization techniques (e.g., [6]) or combining several of the mentioned techniques, such as [31], in which the authors proposed a hybrid approach combining Learning Techniques and Markov Decision Process Planning. In other publications, the optimization technique depends on the specific concerns targeted for planning or the system design integrates different optimization techniques on different layers of the adaption (e.g., for global versus local optimization) [32, 33, 34, 35]. Further, optimization techniques for adaptation planning are often influenced by dynamic environment handling theories as game theory [36], control theory [37], and graph theory [38]. A reason for this could be that adaptive systems have to react on changes in their environment. Accordingly, those systems must take the influence of activities of other systems in the shared environment into account for an optimal adaptation decision. The mentioned dynamic environment handling theories supports this.

Figure 4 provides an overview of the absolute frequency of papers per optimization technique. The investigations have shown that Learning Techniques and Genetic Algorithms are the most frequently used optimization techniques accounting for 19.5% and 17.9%, respectively, and as a result, the authors of this work conclude that these are the most important types of optimization techniques for adaptive systems. Further important techniques are Integer Programming (8.1%), Markov Decision Process Planning (8.1%), Greedy Algorithms (7.3%), Heuristic Algorithms (7.3%), Stochastic Optimization (7.3%), Convex Optimization (6.5%), Metaheuristic Algorithms (4.9%) and Nonlinear Programming (8.9%). Miscellaneous techniques that could not be assigned to one of those categories were summarized as “Others” accounting for 8.9%.

Additionally, Figure 5 provides a detailed overview of optimization techniques in application domains. We observed that cloud computing (22.50%), web services (14.63%), multi-robot systems (10.83%), autonomous vehicles (8.33%), energy provision (6.67%), commerce (6.67%), and intelligent traffic management (5.83%) are the most

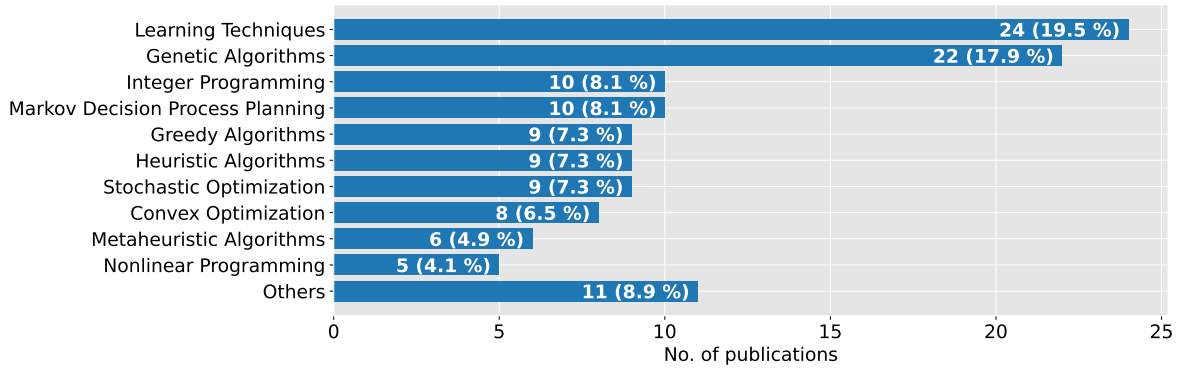

Figure 4: Results of the literature review – Frequency of optimization techniques found in 115 papers. As there were a few publications (namely [6, 31, 32, 33, 34, 35, 39, 40]) which we categorized into several techniques, the total number of counts is 123.

dominant applications areas. These areas are typical domains for adaptive systems due to their dynamic environments. Further, we clustered several less present applications: (i) energy provision includes smart grids (5.83%) and device power management (0.83%); (ii) commerce subsumes e-commerce (4.17%), a travel reservation system, and task allocations in supply chains and within social networks (0.83% each); (iii) health services composed of remote health services and regional emergency management+(1.67% each); (iv) security services represented by mobile application reconfiguration and security management (0.83% each); and (v) smart city includes smart cities in general and shared mobility (0.83% each). Further, some use cases have not been described in detail; hence, these approaches were counted as abstract domains. In [45], the authors present a solution for the optimization/coordination of adaptations in multi-agent systems; however, the authors do not describe a specific application. Additional, in three publications authors apply there approaches within service-oriented architecture, but also abstract service definition rather than a specific service-oriented system [42, 46, 47].

However, it is not possible to determine or identify the over-proportional use of an optimization technique in a specific domain. Additionally, as the number of approaches is in general rather small and having a diverse distribution for the application domains we will discuss details of the application domains only for Learning Techniques (see Section 4.1) and Genetic Algorithms (see Section 4.2) and rather focus in this paper on an analysis of the details of the approaches for planning as optimization. In this section, we present the publications of these categories and name important fields of application and optimization targets. The first two subsections (Section 4.1 and Section 4.2) cover the most two relevant optimization techniques – Learning Techniques and Genetic Algorithms – including a more detailed description of the application domains, while Section 4.3 covers all other techniques.

4.1. Learning Techniques

Although machine learning is actually not an optimization technique, we include the Learning Technique approaches Reinforcement Learning (RL) and Learning Classifier Systems (LCS) since those target optimization [48]. For that reason, we observed that Learning Techniques are the most applied optimization techniques used for planning in adaptive systems (see Figure 4). Our analysis reveals that approximately 70% of the identified approaches that use learning-based optimization apply RL techniques. This seems not to be surprising, as those approaches can cope with the dynamics in the environment of adaptive systems due to their iterative nature. 5 approaches (approx. 22%) apply statistical / machine learning techniques, mainly artificial neural networks; 2 approaches (approx. 8%) use LCS. Still, one has to keep in mind, that we target in this paper planning as optimization for adaptation planning, not learning itself. We identified the presence of the following application domains in our literature review: cloud computing (30.8%), multi-robot systems (23.1%), intelligent traffic management (11.5%), web services (11.5%), e-commerce (7.7%), games (7.7%), intelligent workflow management (3.8%), and smart grids (3.8%). On the one hand, those are typical system domains for adaptive systems. On the other hand, the application of the mentioned learning techniques, especially RL, was often successfully applied in those domains. Reasons for that are that the domains can be easily simulated (e.g., gaming, traffic management, or robots) or those support a clear mapping of adjustments in the system to the resulting performance (e.g., cloud computing, web services, or e-commerce).

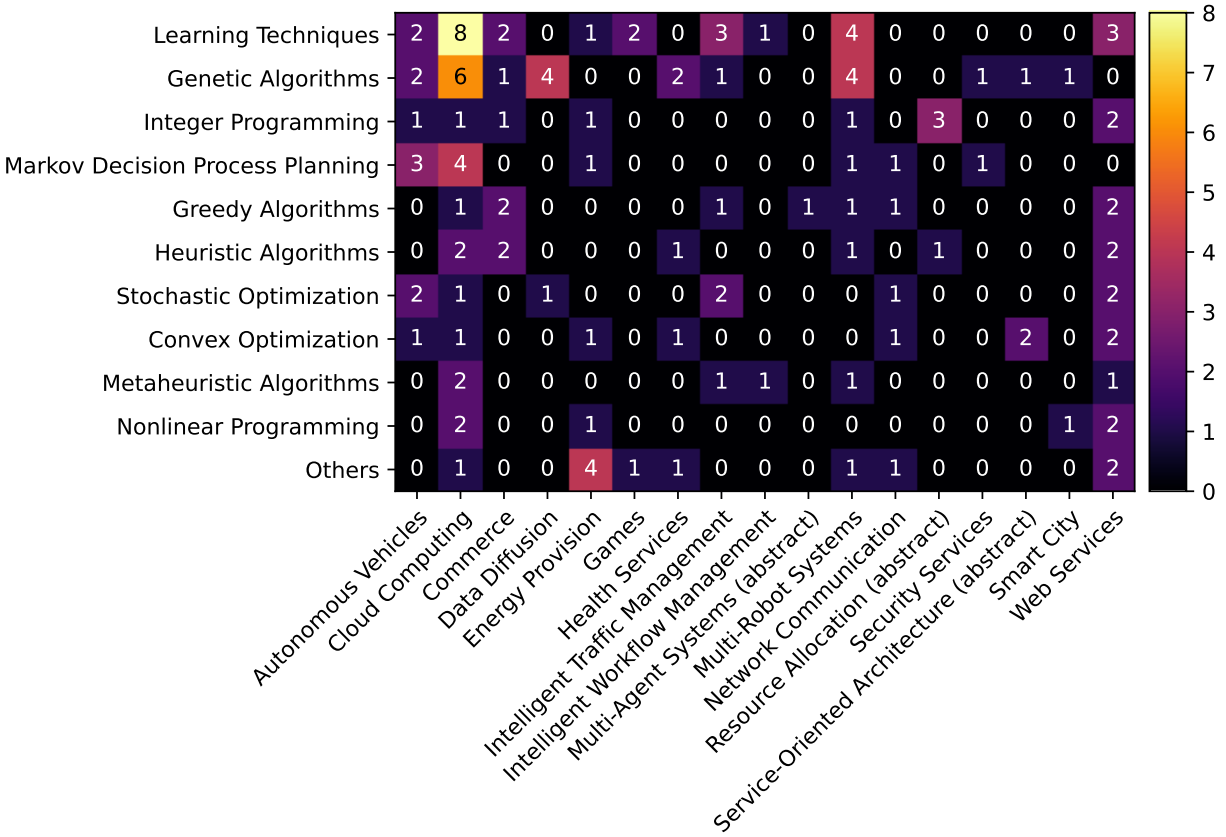

Figure 5: Results of the literature review – Mapping of optimization techniques vs. application domains found in 115 papers. As there were a few publications (namely [6, 31, 32, 33, 34, 35, 39, 40]), which we assigned to several techniques, and some of the publications (namely [31, 41, 42, 43, 44]) applied their approaches in several domains, the total number of counts is 129.

RL methods, such as Q-Learning [49], are rather an universal optimization technique but suitable in adaptive systems to achieve (near-)optimal control. The system learns optimal actions for each environment state using feedback and the action history. Convergence for this technique can be shown for many domains, but not for all real-world applications [50]. The concept of Markov Decision Processes is a foundation for decision-making in adaptive systems. RL can be understood as a solution to problems modeled as such processes. Therefore, some RL approaches make use of Markov Decision Processes. Moreover, this theoretical framework can be extended to more practical cases using (Decentralized) Partially Observable Markov Decision Processes [51].

As primary optimization targets, organization procedures in distributed systems (e.g., task allocation [52, 53] and coalition formation [34, 54, 55]), as well as the handling of environment changes [56, 57, 58], evolved. Distributed systems or applications are characterized by the fact that adaptations or performed actions influence the environment or surrounding systems. These effects are often hard to predict at design time. Hence, RL is suitable in such cases because runtime feedback for performed actions is used to derive future actions. For instance, Wang et al. [56] present an RL-based approach, how a service-oriented and adaptive web application can handle changes in the Quality of Service of connected services. More general machine learning approaches address the control of workload distribution [59] and fault detection and diagnosis in distributed systems [60].

Examples for applying RL for planning and decision-making in adaptive systems are [36, 51, 61, 62, 41]. At this point, the works [61] and [31] are particularly worth mentioning. Kim and Park [61] focus on online planning, where relationships between system configuration and environment settings are learned at runtime, while Pandey et al. [31] propose a hybrid planning approach using a LCS to decide whether a quick reactive adaption or a deliberative adaption is to be preferred. In general, LCS or Learning Techniques are suitable for generating and maintaining adaptation

rules at runtime, as shown in the works of Kramer and Karl [63], Zeppenfeld and Herkersdorf [64], and Zhao et al. [65], respectively. Moreover, Learning Techniques can also be used in the fields of policy-focused systems [66], practical applications like auto-configuration of virtual machines [67], and resource allocation in data centers [68]. RL techniques are also a natural choice for multi-agent or multi-robot navigation [69] or collaboration [58, 70], as interactions and useful behavior can be learned at runtime.

4.2. Genetic Algorithms

Genetic Algorithms (GA), introduced by Holland [71], are suited in cases when some of the design variables are continuous and others are discrete, as well as when the design space is discontinuous and non-convex [13]. Genetic or more general evolutionary algorithms or programming techniques can be used to optimize the planning of adaptive systems. Like Learning Techniques, they are in particular present in self-organization and coordination in distributed systems [72, 73, 74, 75]. Genetic Algorithms are applied in cloud computing (26.1%), data diffusion (17.4%), multi-robot systems (17.4%), autonomous vehicles (8.7%), e-commerce, intelligent traffic management, mobile application reconfiguration, regional emergency management, remote health service, smart city, and an abstract service-oriented architecture (4.4%). Especially, to illustrate autonomous organization based on Genetic Algorithms, different authors select network operation as an example application [76, 77, 78]. This finding is in line with the "traditional" optimization approaches, e.g., vehicular routing or graph-based network optimization, such as in manufacturing. Approaches concerning the planning, decision-making, and reconfiguration of adaptive systems as main targets are given by several examples [79, 80, 81, 82, 83, 84, 85]. Both application scenarios are characterized by a large design space and a complex optimization process. Genetic Algorithms are an excellent choice in these areas, as traditional mathematical optimization techniques would need a lot of computing resources and time for the solution process. They are able to deliver near-optimal solutions in these application scenarios while consuming much fewer resources.

Further usages of Genetic Algorithms include runtime testing [86] and workflow optimization [87]. The works [6] and [88] are focused on the reaction of an adaptive system to sudden changes in its environment. Kinneer et al. [89] propose an approach using reusable repertoires of adaption strategies to improve planning effectiveness. The paper of Andrade et al. [90] examines architectural design aspects with regard to feedback loop control. Therefore, *NSGA-II*, a commonly used evolutionary algorithm, is used. Caldas et al. [33] use NSGA-II to optimize a strategy manager and a strategy enactor by finding the optimal configuration. Another widely used algorithm of this type is *SPEA2*, which has been applied to adaptive systems by Kinneer et al. [84]. We identified that most approaches (37.04%) implemented their own version of a genetic algorithm. From the available "out-of-the-box" algorithms / implementations, the already mentioned approaches NSGA-II (14.81%) and SPEA2 (7.41%) are most commonly used together with novelty search (7.41%). Besides, several approaches are used once: clonal plasticity, D-STM, IBEA, GPM, ISL, 1+1 ONLINE EA, fish schools, Bayesian optimization, and artificial bee colony. As one can see, around one third of the approaches integrate an individual version of a Genetic Algorithm. There are arguments for and against such an individual implementation. It seems to be a rational choice as it provides the possibility to adjust the optimization procedure to the specifics of the underlying use case, e.g., omit mutations and highlight cross-overs if those are more applicable. However, this comes with the likelihood that implementations errors might occur. "Out-of-the-box" approaches provide the advantage to use an existing algorithm or even an implementation (e.g., provided as a specific library), which simplifies the integration of the approach but also facilitates the mapping of the specific components and parameters of the use case system to the task of optimization as planning.

4.3. Further Optimization Approaches

In this section, we describe the remaining categories shown in Figure 4. Some of the mentioned categories might often be included in others (e.g., Markov Decision Process as part of Stochastic Optimization [6]), but we want to point out the intensified number of use cases.

Integer Programming. An optimization problem where all components of the design vector are restricted to discrete (integer) values can be solved with Integer Programming (IP) techniques. If some of the design variables are discrete, other continuous, Mixed Integer Programming can be applied. In adaptive systems and their applications, some problems may be formulated roughly as *"How many?"*-questions, e.g., *"How many servers do I have to provide?"*, which have only integer answers and thus, might be solved with IP. As a standard form, an Integer Programming problem can be expressed as a constrained optimization problem: Find x, which maximizes $f(x) = c^T x$,

subject to $g_1(x) = Ax \leq b$ and $g_2(x) = x \geq 0$, where $x \in \mathbb{Z}^n$, $c^T \in \mathbb{R}^{mxn}$, $A \in \mathbb{R}^{rxn}$ and $b \in \mathbb{R}^r$. In practical applications, the *IBM ILOG CPLEX*[1] optimization framework or similar tools are often used to solve this category's problems. For instance, we find approaches of this type in service or feature selection [91, 92] and load balancing or distributing applications [37, 40]. Moreover, the works of Wu et al. [93, 94] use Mixed Integer Programming for optimal resource allocation in multi-agent systems and show how tasks can be decomposed into phases. Feo Flushing et al. [95] focus on optimal decentralized task allocation in multi-robot systems and use Mixed Integer Programming for adaptive re-planning at runtime. Other usages of IP can be found in the fields of resource scheduling [96] and item assignment in general [97] in adaptive or multi-agent systems.

Markov Decision Process Planning. Markov Decision Processes (MDP) are commonly used for modeling decision-making and planning tasks in adaptive systems. In general, they describe discrete-time stochastic control processes. In the fields of adaptive systems, they are used to model and optimize decision-making with probabilistic environment behavior and/or uncertainty [98, 99, 100, 101, 102]. As an advantage of this approach, Markov Decision Processes can be used as offline planning support tools and might be combined with other strategies (e.g., deterministic [98] or learning-based planning [31]). To solve optimization problems based on Markov Decision Processes, often dynamic programming techniques are suitable. For example, Angelidakis and Chalkiadakis [103] use a dynamic programming algorithm based on value iteration for an optimization problem in power distribution networks. Other applications take place in the field of autonomous vehicles. Basich et al. [104] optimize the decision-making through human feedback, while Bouton et al. [34] address the decision-making in pedestrian collision avoidance. Scheerer et al. [105] focus on validating adaption strategies at design time to ensure system performance.

Greedy Algorithms. Greedy Algorithms have numerous applications in different fields of computer science. Many Greedy Algorithms appear in graph-theoretic problems, e.g., the algorithms of Kruskal [106] and Prim [107] for solving the minimum spanning tree problem. In general, they follow the heuristic to select the (locally) optimal solution out of a candidate set at each iteration. As an advantage, Greedy Algorithms terminate after a reasonable number of steps and approximate at least a local optimum in most cases, especially if a global optimum is hard to compute. Hence, they often reach an acceptable solution with limited resource consumption. However, the quality of the result and convergence speed often depends on the starting value. Often, a supporting heuristic is used to determine a suitable initialization. Distributed systems are generally an application area with these characteristics. Especially relevant for practical or industrial use cases are hereby optimization algorithms in the fields of task allocation [108, 109] and resource allocation [110, 111]. Further applications of Greedy Algorithms can be found in the areas of computing clusters [112] and web databases [113]. Fritsch et al. [114] show that Greedy Algorithms can also be used for scheduling adaptations in cases when these adaptations are time-bounded or underlying other constraints. As mentioned before, Greedy Algorithms are often used for solving graph-theoretic problems. Escoffier et al. [45] present greedy solutions based on graph theory, which can be applied in the fields of adaptive systems.

Heuristic Algorithms. Heuristics try to guide a way through the search space to find an optimal solution faster than classical mathematical optimization methods. In contrast to Metaheuristics, they are domain-specific. Therefore, the concrete appearance is highly dependent on the application scenario. We use *Heuristic Algorithms* as a generic term in this work to indicate that the optimization process is based on a domain-specific heuristic. In some cases, a Heuristic Algorithm does not guarantee to find a globally optimal solution. This is often caused by the use of random variables, non-deterministic behaviors, or the choice of starting values. However, Heuristic Algorithms are suited for problems when a globally optimal solution is expensive to compute and/or a solution is required within a bounded time interval. Our investigation has revealed that Heuristic Algorithms are mostly used in the fields of resource provisioning, allocation, and planning [32, 115, 116, 117, 118] as well as coalition formation [119]. In these scenarios, the reason for this is that application-specific characteristics can be added to the algorithm and improve or accelerate the optimization process. Further practical use cases of Heuristic Algorithms can be found in the fields of supply chains [120] and web search engines [121]. Cooray et al. [122] introduce a Heuristic Algorithm for self-adaptation to increase the reliability of an adaptive mobile emergency response system.

Stochastic Optimization. In real-world applications, adaptive systems often deal with non-deterministic quantities, like environment variables or even internal parameters. Stochastic Optimization techniques deal with such cases, where the use of random or stochastic variables is mandatory, and uncertainty is present and not negligible. In general, adaptive systems might use stochastic programming or often Bayesian optimization for decision-making and

[1] `https://www.ibm.com/analytics/cplex-optimizer`

control [6, 123, 124, 125, 126, 127]. For instance, Esfahani et al. [128] focus on decision-making for self-adaptive software in the presence of uncertainty, as the work of Mikic-Rakic and Medvidovic [35] describes a way of handling downtimes of dependent software components. Palmerino et al. [129] consider tactic volatility using multiple regression analysis and autoregressive integrated moving average.

Convex Optimization. Convex Optimization approaches target problems where the objective function is a convex function[2], and the design space is a convex set[3]. Many solution methods for these problems are based on linear or quadratic programming. In the fields of adaptive systems, Convex Optimization techniques are often used in general control mechanisms [121, 130, 43, 131]. They have applications in cloud environments [132] and service-oriented architectures [46]. Javed and Arshad [40] use a linear programming based algorithm and case-based reasoning for self-optimization and evaluate their framework on an electricity distribution system.

Metaheuristic Algorithms. In contrast to Heuristics, Metaheuristics are domain-independent. Metaheuristic Algorithms often perform a local or neighborhood search, which means using a given solution X_1, a better solution X_2 in the neighborhood[4] of X_1 is searched. This limits the computational effort, but the algorithm might not guarantee to find a globally optimal solution. Similar to Greedy Algorithms, these approaches are suitable in applications where time-bounded reactivity or limited computational resources are challenges. For instance, Zhang et al. [133] use simulated annealing for task allocation in a distributed system. Other variants of local search found in adaptive systems literature include hillclimbing [134], neighborhood search [38, 135], and tabu search [136, 137].

Nonlinear Programming. In general, Nonlinear Programming (NLP) techniques are suitable if the solution of an optimization problem cannot be determined analytically. Therefore NLP techniques are applied in constrained optimization problems, where the constraints are not explicit functions of the design vector [13]. Lama and Zhou [138] use complex, non-differentiable objective functions solved by a pattern search algorithm for optimizing automated resource allocation in cloud environments. Jung et al. [139] use offline gradient-based optimization as a basis of adaptation policy generation. Other applications of nonlinear optimization approaches can be found in the fields of power management [140], decentralized planning [44], and web services [141].

Miscellaneous Approaches. In this category, we describe approaches, which cannot be assigned clearly to one of the categories above. For instance, Chuang et al. [142] use hierarchical fuzzy systems to optimize the Quality of Service of mobile adaptive software. Lee and Fortes [32] use fuzzy logic to control the number of concurrent jobs in big-data analytics. Control theory principles and techniques are utilized in the work of Caldas et al. [33] for adaptive performance control. Wang et al. [143] expand this utilization to power management. Other optimization approaches include combinatorial optimization [144, 145, 146], weighted sum search [147], and distributed constraint optimization [148, 149, 150], which is a general problem representation framework widely used for multi-agent systems. As we are interested in providing a complete picture of the landscape rather than a detailed discussion of each approach, we do not discuss these approaches that do not fit in one of the optimization categories.

5. Planning Optimization in Dynamic, Competitive, and Distributed Environments

As presented in Section 3, we included 115 papers from our initial set of 9,588 papers. In the previous section, we presented the set of identified optimization techniques that are applied for adaptation planning in self-adaptive systems. In this section, we detail our discussion and focus on the specific aspects: anytime learning (see Section 5.1), multi-objectiveness (see Section 5.2), distributed optimization (see Section 5.3), and situation-awareness for switching the optimization techniques (see Section 5.4). This answers the research questions *RQ2*, *RQ3*, *RQ4*, and *RQ5*. Figure 6 provides an overview of the presence of papers concerning the topics anytime learning and distributed optimization over the relevant time frame.

5.1. Anytime Learning

The possibly rapid and frequent changes in the environment of a self-adaptive system might significantly decrease the available time for an optimization technique for adaptation planning. This means that those techniques sometimes

[2] A function $f(x) : D \to \mathbb{R}$ is called convex if $f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2)$ for all $x_1, x_2 \in D$ and all $\alpha \in [0;1]$, where D is a convex set

[3] A set D is called convex, if $(1-\alpha)x_1 + \alpha x_2 \in D$ for all $x_1, x_2 \in D$ and all $\alpha \in (0;1)$

[4] Therefore a neighborhood relation on the design space D must be defined, e.g., using a norm $\|\cdot\|$ and a threshold ϵ: *A solution* X_2 *is a neighbor of solution* X_1*, if* $\|X_2 - X_1\| < \epsilon$

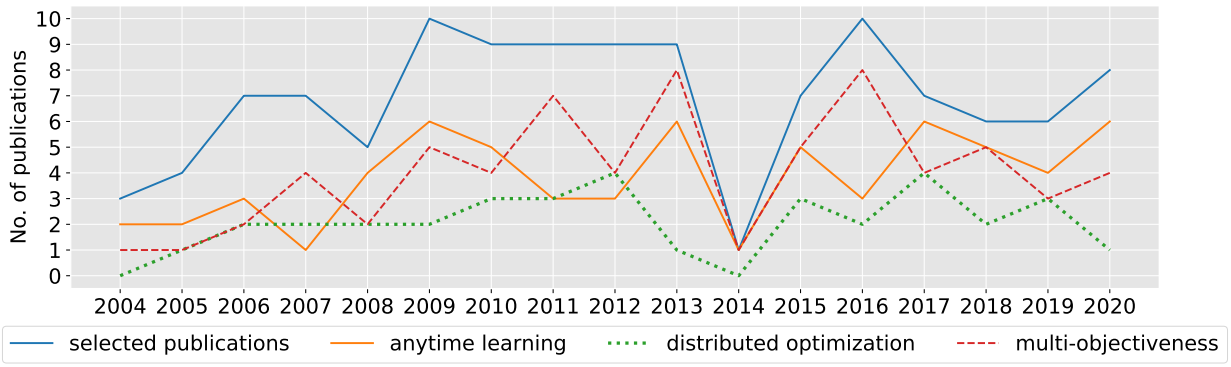

Figure 6: Results of the literature review – Number of all included publications in this survey, number of publications supporting anytime learning, supporting distributed optimization, and supporting multi-objectiveness per year

need to focus on searching for fast and "good enough" solutions rather than for "optimal" ones. Hence, self-adaptive systems need to focus on anytime learning algorithms that can provide intermediate solutions and do not have to wait until the optimization process finally reaches a (successful) end [7]. From an implementation point of view, this can be achieved using anytime algorithms for optimization [8]. Accordingly, the second research question addresses the presence of anytime learning in the identified set of literature:

RQ2 - To what extent is anytime learning supported?

When investigating the existing literature regarding anytime learning in self-adaptive systems, we identified that 56.1% of the studied works (69 publications) integrated approaches that support anytime learning. The papers show a diverse set of used optimization techniques. The most common techniques that support anytime learning include Genetic Algorithms (21 papers), Learning Techniques (16 papers), and (Meta)Heuristics (12 papers). Applied techniques include (but are not limited to) SPEA2, Stochastic Search, NSGA-II, Q-Learning, Markov Chains, Multi-agent RL, and Greedy Algorithms. Table 3 shows an overview of the application of anytime learning.

Table 3: Overview of anytime learning used within optimization techniques.

Optimization Technique	No. of publications		References
	total	anytime	
Learning Techniques	24	16	[31][34][36][51][52][54][56][57][59][61][63][65][66][67][69][70]
Genetic Algorithms	22	21	[33][72][73][74][75][76][77][78][79][80][81][82][83][84][85][86][87][88][89][90][42]
Integer Programming	10	0	
Markov Decision Process Planning	10	4	[31][34][101][104]
Greedy Algorithms	9	3	[110][112][114]
Heuristic Algorithms	9	6	[32][115][116][119][120][122]
Stochastic Optimization	9	2	[127][129]
Convex Optimization	8	1	[130]
Metaheuristic Algorithms	6	6	[38][133][134][135][136][137]
Nonlinear Programming	5	4	[139][140][44][141]
Others	11	6	[32][33][144][145][146][150]

We already argued that anytime learning [7] techniques might be beneficial as those techniques constantly provide a solution for adaptation planning in contrast to exact algorithms, such as Integer Programming. Consequently, anytime learning can increase the robustness of the system as those techniques provide intermediate solutions which can fasten the reaction to changes rather than waiting until the final solution is found, which is the case for exact solutions.

On the other hand, anytime learning techniques might have disadvantages that system designers and developers have to take into account. As those techniques do not explicitly search for an exact solution, hence, the globally

Table 4: Overview of optimization techniques supporting multi-objectiveness. All approaches not supporting multiple objectives are single-objective approaches.

Optimization Technique	No. of publications		References
	total	multi-obj.	
Learning Techniques	24	17	[31][34][36][51][52][54][55][56][57][62][41][64][65][66][67][69][70]
Genetic Algorithms	22	15	[6][76][77][78][79][80][82][83][84][86][87][88][89][90][42]
Integer Programming	10	6	[37][92][94][95][97][153]
Markov Decision Process Planning	10	8	[31][34][99][100][101][102][103][105]
Greedy Algorithms	9	5	[108][110][112][113][45]
Heuristic Algorithms	9	5	[117][118][119][121][122]
Stochastic Optimization	9	3	[6][123][125]
Convex Optimization	8	6	[130][43][131][46][47]
Metaheuristic Algorithms	6	1	[136]
Nonlinear Programming	5	1	[44]
Others	11	4	[142][143][146][147]

optimal solution, it might be possible that only a "good enough" solution is found and the optimizer returns a local optimum. Additionally, one has to take into account that the result of the optimizer has to be mapped to a system configuration. The iterative nature of the algorithms might lead to constant adjustments in the system, which can influence the system performance and in the worst case also lead to adaptation oscillation [151]. Often, it might be the case that there is a direct mapping, i.e., the configuration parameters are used in the optimization. However, this is not always the case. Either way, as each newly identified solution of the optimization process triggers a new adaptation solution. If this is frequently the case, the system might suffer from adaptation oscillation [151]. This means that the system configuration is frequently adjusted through adaptation, resulting in undesired behavior, such as the user getting irritated when the system behavior constantly changes.

Consequently, we propose to identify and work on approaches that combine exact techniques with anytime learning. This way, it might be possible to run both optimizations in parallel and – in case the exact algorithms cannot return a solution fast enough – to have the anytime learning results as a backup solution, e.g., in situations where the system is running without negative impacts and the adaptation focuses on further improving the system performance. Those approaches might also be highly interesting in distributed settings by having a central instance for the exact optimization and many decentralized ones that react to small changes by applying anytime learning techniques.

5.2. Multi-objectiveness

In general, objective functions can be single- or multi-objective. Regarding multi-objectiveness, Pareto optimality is a highly relevant concept. Pareto optimality is defined as "analytic tool for assessing social welfare and resource allocation [where an] allocation is considered Pareto optimal if no alternative allocation could make someone better off without making someone else worse off" [152]. Transferred to the optimization of adaptive systems, this means that it is not possible to achieve a globally optimal setting through increasing the utility of one objective if the utility of one or several of the other objectives gets decreased. For a self-adaptive system with its large amount of subsystems and heterogeneous user groups, multi-objective settings are highly relevant, especially to find cost-benefit trade-offs regarding the system performance. Hence, we formulated a third research question:

RQ3 - To what extent is multi-objectiveness supported?

Table 4 shows the results of our analysis for RQ3. We observed that around 57% of the identified approaches (71 approaches) target multi-objectiveness. This seems natural for the characteristics of self-adaptive systems. Especially for Learning Techniques (17 papers) and Genetic Algorithms (15 papers), multi-objective approaches are frequently present. However, the results show that with the exception of Stochastic, Metaheuristic, and Nonlinear Programming approaches, more than half of the identified approaches are multi-objective in all other categories.

Table 5: Overview of the usage of distributed optimization techniques.

Optimization Technique	No. of publications		References
	total	distributed	
Learning Techniques	24	11	[34][36][51][52][53][56][57][58][66][69][70]
Genetic Algorithms	22	5	[72][74][77][78][42]
Integer Programming	10	0	
Markov Decision Process Planning	10	1	[34]
Greedy Algorithms	9	3	[108][109][111]
Heuristic Algorithms	9	4	[32][117][119][120]
Stochastic Optimization	9	1	[129]
Convex Optimization	8	1	[47]
Metaheuristic Algorithms	6	3	[133][136][137]
Nonlinear Programming	5	2	[44][141]
Others	11	5	[143][145][148][149][150]

As mentioned, multi-objectiveness is especially relevant for evaluating possible adaptations, i.e., system setting, with regard to their cost-performance ratio. However, as those optimizations often return Pareto optimal solutions, potentially many different solutions might have the same impact with regard to the global utility. This complicates the choice of the system configuration. Further, sometimes single-objectiveness is achieved as the relation of different important parameters, which might represent objectives, are expressed differently, e.g., by a weighted utility function. The definition of such a utility function is highly complicated and requires a lot of domain knowledge. Especially, the definition of the weights might be challenging and use case-specific. Additionally, the definition of objectives requires the definition of usable metrics to operationalize those objectives. Optimization functions will use those metrics to calculate the utility of an identified solution. The definition of those metrics can be another challenge.

Multi-objectiveness requires a trade-off of different objectives. Often, this leads to Pareto optimal solutions, as it is often not possible to optimize several goals simultaneously. One solution can be approaches that focus on many-objectiveness – which refers to the optimization tasks involving several (conflicting) objectives to be optimized concurrently [154]. Such approaches enable also to integrate user-specific, differing goals. Further, such approaches might also support situation-awareness, as depending on the situation one or another objective can be favored.

5.3. Distributed Optimization

Many self-adaptive systems are distributed [10]. First of all, mostly the adaptation logic which controls adaptation is encapsulated from the managed subsystem [9]. Additionally, many adaptive systems are often integrated into a composition of different (sub)systems. Such system are naturally acting in a distributed fashion, either cooperatively or competitively, e.g., if they are (maybe implicitly) competing for resources in a shared environment. In such settings, particularly in cooperative settings, a distributed optimization might be beneficial. This can be achieved by either distributing the decision-making or achieving a global optimization through a central planner. However, such a central planer has to achieve a trade-off between local objectives and needs to take local constraints into account. Hence, we formulated the following research question which especially targets those adaptive system from our identified literature that are distributed:

RQ4 - To what extent are distributed optimization techniques integrated?

In our literature analysis, approximately 30% of the identified papers rely on distributed optimization (36 papers). One has to note that distributed optimization approaches are only present for distributed adaptive systems, e.g., CAS. This includes local (non-coordinated) adaptation planning, decentralized optimization techniques, i.e., multi-agent RL, as well as hybrid planning, e.g., combining a central optimization with local adjustments through the application of a local optimizer or a learning mechanism. Table 5 shows that the large amount of distributed optimization uses Learning Techniques (11 papers). Further, Genetic Algorithms (5 papers), (Meta-) Heuristic Algorithms (7 papers), or Greedy Algorithms (3 papers) were frequently used for distributed adaptation planning as optimization.

Furthermore, central optimization could deliver a global optimal solution. Such central decision-making would be also possible in distributed adaptive systems. However, it comes with the costs for collecting the required data from the local subsystems. Additionally, it might introduce a single point of failure and, especially in large systems, the decision-making can be complex. The mentioned decentralized optimization techniques help to distribute the workload for adaptation control. This local decision-making also takes local constraints into account.

Related to local constraints is the achievement of fairness. When focusing on global concerns only, it might be possible that single instances are disadvantaged for the sake of global utility. Accordingly, mechanisms for achieving fairness and concerning local constraints might be important. Alternatively, it is possible to have degrees of freedom that the instances can locally optimize. This results in a hybrid optimization approach which integrates macro-level planning under longer time requirements with micro-level (local) decisions that obey local objectives and constraints. Such an approach would result in (i) increased robustness as adaptation decision can obey local constraints, (ii) an improved utility of both the autonomous subsystems and the overall system, (iii) and a fast adaptation to changes in the characteristics of the optimization problem (e.g., in terms of concept drift/shift). However, implementations of such approaches cannot be found in literature yet.

5.4. Situation-awareness

In [6, 12], we showed that (i) different optimization techniques might be superior depending on the characteristics of the situational characteristic or that (ii) different algorithms might deliver the same quality with respect to the objectives but might have different performance implication (e.g., faster computation or less required memory). This conforms to other observations: It will not be feasible to define the best-fitting strategy for each situation [155] as those situations are determined by various parameters, each having at minimum two possible manifestation (in case of binary value) up to an incredible large number (e.g., in case of 64 bit numbers). Testing all those configuration options is not possible [4]. Additionally, according to the "*No-Free-Lunch-Theorem*" from 1997 [156], there is no general optimization method that performs best in all scenarios. Instead, expert knowledge is needed to decide which optimization method to choose for a specific situation with its own (data pattern) characteristics. Accordingly, in such systems it seems beneficial to link the choice of the optimization technique to the current situation, i.e., switching the adaptation technique and/or adjusting the parameters of a technique (e.g., through hyper-parameter tuning). Hence, we further address with our literature study the following research question:

RQ5 - Does the used techniques for planning as optimization support situation-awareness?

However, we identified only two approaches that provide a situation-aware switch of the optimization technique. AdOpt [40] provides "an adaptive self-optimization approach which uses multiple optimization techniques to incorporate a resilient self-optimization in a given system". Instead of directly planning the adaptation, the planner in AdOpt first identifies the appropriate optimization technique for the given system state and then generates a runtime model for adaptation planning as optimization. The authors of [31] provide a hybrid planning approach that "can combine reactive planning (to provide an emergency response quickly) with deliberative planning that takes time but determine a higher-quality plan". As fast, reactive planning might also deliver adaptation plans to potentially decrease the quality of the system's performance, the challenge in this approach lies in deciding when to use which planning approach. In the paper, the authors present several learning-based strategies for this decision.

We want to stress that we focus on the switch of the optimization technique or its parameters within this section. Additionally, several works (e.g., [61], [63], [118], or [121]) combine different optimization techniques in hybrid approaches. This might be beneficial for fine-grade planning or combinations of different time horizons for planning, e.g., proactive planning with reactive planning as a backup. Further, it might be used for situation-aware planning as optimization as well. However, none of the identified publications do so. Hence, we did not include this perspective when we talk about situation-awareness.

6. Threats to Validity

We used a well-structured approach for our literature review to provide a structured analysis and eliminate bias in the process. However, some threats to validity still exist, which we discuss in the following. Each paper was analyzed

by one of the authors of this publication. As humans are involved, the presence of subjective bias cannot be entirely excluded. To limit this risk, we double-checked each analysis by at least a second reviewer for each paper.

The choice of the venues might be biased. We rely on discussion with experts to identify the relevant venues. Still, it might be possible that other conferences and journals provide relevant work that we did not take into account. However, as mentioned in Section 3, the application of a term-based literature search is complicated as the term "optimization" is often used in another context. Thus, to achieve high quality and still following a structured search process, we limit the set of possible papers to the list of venues.

Even if the number of included works from IEEE Software and TOSEM is very low, we think that it is still necessary to consider those journals as several publications of the field of self-adaptive systems originate in those journals, even if the actual number of publications that target optimization as planning is very low. We decided to not include additional general purpose journals (such as IEEE Access) as we do not want to widen the scope too much for keeping the focus on the important venues of our targeted system domains. We acknowledge that this can be a threat to validity, but we act in line with other review papers in the field (e.g., [30]).

Similarly, through reducing the search to a defined list of venues, we also have implicitly covered a set of concepts that are related to self-adaptive systems. This coverage is rather extensive. However, it might be possible that a specific system domain is not taken into account.

Further, we focused on optimization techniques for adaptation planning. However, we included techniques based on RL and LCS as those are optimization-based, iterative approaches following the taxonomy from [13]. Accordingly, some might argue that the border towards techniques that fall into the category of machine learning might be blurred here. Others might argue that machine learning techniques also support planning as an optimization idea. However, we clearly focus on the mathematical and stochastic optimization procedures in this paper and clearly distinguish them from the classical application of machine learning with classification and clustering, which is mainly used in self-adaptive systems for analyzing.

Additionally, we omit *search-based software engineering* (*SBSE*) approaches used in adaptive systems for adaptation planning. SBSE [157] aims at applying search-based Metaheuristic techniques to software engineering. Search techniques such as genetic programming examine large search spaces of candidate solutions to find a (near) optimal solution to problems concerning requirements, design, or testing [158]. Traditionally, SBSE is used at design time. Contrary to the traditional approach, dynamic SBSE applies the principle of SBSE at runtime to determine the most suitable system configuration during the planning phase of self-adaptation [159, 160]. Approaches that use dynamic SBSE in self-adaptive systems can be found in the literature (e.g., [130, 134, 161, 162, 163, 164]). However, as we follow the classification of optimization from [13], we did not include SBSE as a specific category.

7. Related Work

This survey connects methods from optimization processes with the application areas of adaptive systems and studies the use of optimization techniques in self-adaptive systems. To the best of our knowledge, there is no survey, which discusses this combination of issues explicitly and to the full extent. In this section, we provide an overview of related surveys from the area of adaptive systems.

There are many literature reviews and summaries in the research field of adaptive systems, which focus on either the general topic or special system aspects. Salehie and Tahvildari [16] broach the issue of self-adaptive systems as such, provide a taxonomy and describe possible realizations of adaptation actions. Further introduction material and reference work in the area of adaptive systems is provided by Krupitzer et al. [2], Macias-Escriva et al. [165], or more recently by Wong et al. [166].

The work of Weyns [167] provides an overview of adaptive systems with a focus on software engineering aspects. Findings in the fields of task automation, architecture-based adaptation, runtime models, goal-driven adaptation, uncertainty management, and control-based adaptation are summarized. It is concluded that control theory can act as a theoretical foundation for adaptations and decision-making. The surveys of Patikirikorala et al. [168] and Shevtsov et al. [169] take a closer look at this issue.

More closely to the topic of this paper, Saputri et al. [170] presented an overview on the application of machine learning in self-adaptive systems, which is used to handle self-adaptation, but also for analyzing the requirement for adaptation. As an outcome of the 2018 GI-Dagstuhl Seminar "Software Engineering for Intelligent and Autonomous

Systems", D'Angelo et al. discuss the current state of the art for learning in complex self-adaptive systems in [30]. In addition, Gheibi et al. [171] investigated the application of machine learning in self-adaptive systems based on the MAPE feedback loop. Their results showed a major proportion of machine learning taking place in the analysis phase (83 studies), followed by the planning phase (57 studies), the monitoring phase (23 studies) and the execution phase (one study). The authors further revealed that machine learning is mainly used to "update/change adaptation rules/policies" and "predict/analyze resource usage". However, they only focused on machine learning and did not include other techniques to optimize the adaptation planning.

8. Conclusion

This article investigated the use of optimization techniques for adaptation planning in self-adaptive systems. For this, we looked at 115 publications out of 12 selected venues with 9,588 publications in total. Next, we briefly summarize our results of the literature analysis. Afterwards, we present identified research challenges.

As a first result of our literature review, we found optimization methods belonging to 11 categories according to [13]. We found that Learning Techniques and Genetic Algorithms are the most applied techniques for optimization in adaption planning (RQ1). Additionally, we hypothesized that anytime learning can be helpful for adaptive systems, as anytime learning returns intermediate solutions (in contrast to exact algorithms). This better fits the dynamics of the environment and the requirement for fast adaptations. The results comply with our hypothesis: Heuristics (corresponding to anytime learning, e.g., using anytime algorithms for optimization [8]) are more frequently applied than exact algorithms (see RQ2). Multi-objective optimization helps to incorporate various objectives (but also constraints), and also supports the process of adaptation planning in large, distributed adaptive systems. We observed that around 57% of the approaches support multi-objectiveness (see RQ3). As adaptive systems often are distributed [10], we further investigated the presence of distributed optimization. Regardless the benefits of distributed optimization (e.g., the integration of local constraints), distributed optimization can be highly complex, as it might results in local optima which are conflicting. Hence, it is not surprising that distributed optimization are only present in around a third of the approaches (see RQ4). According to the "*No-Free-Lunch-Theorem*" [11] there is no general optimization method that performs best in all scenarios. In the context of adaptive system this requires to potentially switch the optimizer after a change in the context as this triggers a change in the data pattern [6, 12]. Consequently, we focused on situation-awareness as last dimension, which is only supported by two approaches (see RQ5).

Based on our results, we derive several research challenges related to anytime learning , many-/multi-objectiveness, distributed optimization, and situation-awareness that have potential for further research. We identified many anytime learning approaches are present in the literature (see RQ2) as those approaches fit the dynamic nature of self-adaptive systems and their environment. However, as those approaches might also deliver system configurations that have a negative impact of system performance, we propose to work on approaches that combine exact optimization techniques with anytime learning. Most of the identified approaches support multi-objectiveness (see RQ3). We further propose to apply *many-objectiveness* [154] -— which refers to the optimization tasks involving several (conflicting) objectives to be optimized concurrently – to support a user-specific, system-specific, or situation-aware choice of the specific objective technique. Additionally, around one third of the approaches support distributed optimization (see RQ4). We expect that system models for hybrid optimization for adaptation planning—which combine global optimized planning with degrees of freedom with local decision-making— will result in (i) an increased robustness against intentionally wrong or even faulty behavior, (ii) an improved utility, (iii) and a fast adaptation to changes in the characteristics of the optimization problem. However, due to the distribution of the relevant data, those approaches are highly complex. Still, the study of those distributed optimization approaches might be beneficial. In [6] and [12], we showed that different optimization techniques might be superior depending on the characteristics of the situation or might be more efficient in terms of computation. Accordingly, switching the adaptation technique and/or adjusting the parameters of a technique depending on the current situation (see RQ5), e.g., through hyper-parameter tuning, might be desirable. However, this is not well presented yet in literature. We recommend for the future to study such approaches for providing the best optimization technique for a specific situation (which comes with a specific data pattern). In the past, we presented such studies in the area of intelligent traffic management systems [6, 12] and smart health [172] which might be an inspiration.

Acknowledgements

This work is funded by the *Vector Stiftung* within the project "The smart, digitized food supply chain" (P2021-0093).

References

[1] B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, J. Whittle, Software Engineering for Self-Adaptive Systems: A Research Roadmap, in: Software Engineering for Self-Adaptive Systems, volume 5525 of *LNCS*, Springer, 2009, pp. 1–26.

[2] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, C. Becker, A survey on engineering approaches for self-adaptive systems, Pervasive Mob. Comput. 17 (2015) 184–206. URL: http://dx.doi.org/10.1016/j.pmcj.2014.09.009. doi:10.1016/j.pmcj.2014.09.009.

[3] J. O. Kephart, D. M. Chess, The Vision of Autonomic Computing, IEEE Computer 36 (2003) 41–50.

[4] S. Lightstone, Foundations of Autonomic Computing Development, in: Proceedings of the International Workshop on Engineering of Autonomic and Autonomous Systems (EASe), IEEE, 2007, pp. 163–171.

[5] S. Tomforde, C. Müller-Schloer, Incremental Design of Adaptive Systems, Journal of Ambient Intelligence and Smart Environments 6 (2014) 179–198.

[6] E. M. Fredericks, I. Gerostathopoulos, C. Krupitzer, T. Vogel, Planning as Optimization: Dynamically Discovering Optimal Configurations for Runtime Situations, in: 2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2019, pp. 1–10. doi:10.1109/SASO.2019.00010.

[7] J. J. Grefenstette, C. L. Ramsey, An approach to anytime learning, in: Machine Learning Proceedings 1992, Morgan Kaufmann, 1992, pp. 189 – 195.

[8] A. D. Jesus, A. Liefooghe, B. Derbel, L. Paquete, Algorithm selection of anytime algorithms, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, Association for Computing Machinery, New York, NY, USA, 2020, p. 850–858. URL: https://doi.org/10.1145/3377930.3390185. doi:10.1145/3377930.3390185.

[9] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, E. Gjorven, Using Architecture Models for Runtime Adaptability, IEEE Software 23 (2006) 62–70.

[10] B. H. C. Cheng, R. Lemos, H. Giese, P. Inverardi, et al., Software engineering for self-adaptive systems: A research roadmap, in: Software engineering for self-adaptive systems, Springer, 2009, pp. 1–26.

[11] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82.

[12] V. Lesch, T. Noack, J. Hefter, S. Kounev, C. Krupitzer, Towards situation-aware meta-optimization of adaptation planning strategies, in: 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2021, pp. 177–187. doi:10.1109/ACSOS52086.2021.00042.

[13] S. Rao, Engineering Optimization: Theory and Practice: Fourth Edition, John Wiley and Sons, 2009. doi:10.1002/9780470549124.

[14] R. Laddaga, Active Software, in: Self-Adaptive Software, volume 1936 of *LNCS*, Springer, 2001, pp. 11–26.

[15] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, A. L. Wolf, An Architecture-Based Approach to Self-Adaptive Software, IEEE Intelligent Systems 14 (1999) 54–62.

[16] M. Salehie, L. Tahvildari, Self-adaptive software: Landscape and research challenges, ACM Trans. Auton. Adapt. Syst. 4 (2009) 14:1–14:42. URL: http://doi.acm.org/10.1145/1516533.1516538. doi:10.1145/1516533.1516538.

[17] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. A. Müller, M. Pezzè, M. Shaw, Engineering Self-Adaptive Systems through Feedback Loops, in: Software Engineering for Self-Adaptive Systems, volume 5525 of *LNCS*, Springer, 2009, pp. 48–70.

[18] J. Zhang, B. H. C. Cheng, Model-Based Development of Dynamically Adaptive Software, in: Proceedings of the International Conference on Software Engineering (ICSE), ACM, 2006, pp. 371–308.

[19] B. Schmerl, J. Andersson, T. Vogel, M. B. Cohen, C. M. F. Rubira, Y. Brun, A. Gorla, F. Zambonelli, L. Baresi, Challenges in Composing and Decomposing Assurances for Self-Adaptive Systems, in: Software Engineering for Self-Adaptive Systems III. Assurances, volume 9640, Springer, 2017, pp. 64–89.

[20] J. Kramer, J. Magee, Self-managed systems: an architectural challenge, in: Rpoc. FOSE, IEEE, 2007, pp. 259–268.

[21] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, F. Zambonelli, A survey of autonomic communications, ACM TAAS 1 (2006) 223–259.

[22] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, M. Mnif, C. Müller-Schloer, U. Richter, H. Schmeck, Observation and Control of Organic Systems, in: Organic Computing – A Paradigm Shift for Complex Systems, Springer, 2011, pp. 325–338.

[23] S. Kounev, P. Lewis, K. L. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, P. Inverardi, J. O. Kephart, A. Zisman, The Notion of Self-aware Computing, in: Self-Aware Computing Systems, Springer, Cham, 2017, pp. 3–16.

[24] O. Babaoglu, H. E. Shrobe, Foreword from the General Co-Chairs , in: Proceeding of the International Conference on Self-Adaptive and Self-Organizing Systems (SASO), IEEE, 2007, pp. ix–x.

[25] C. Müller-Schloer, H. Schmeck, T. Ungerer (Eds.), Organic Computing – A Paradigm Shift for Complex Systems, Springer, 2011.

[26] M. Weiser, The Computer for the 21st Century, Scientific American 265 (1991) 94–104.

[27] A. Ferscha, Collective adaptive systems, in: Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, UbiComp/ISWC'15 Adjunct, Association for Computing Machinery, 2015, p. 893–895. URL: https://doi.org/10.1145/2800835.2809508. doi:10.1145/2800835.2809508.

[28] J. Webster, R. T. Watson, Analyzing the past to prepare for the future: Writing a literature review, MIS Q. 26 (2002) xiii–xxiii. URL: http://dl.acm.org/citation.cfm?id=2017160.2017162.

[29] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, BCS Learning & Development Ltd., Swindon, UK, 2008, pp. 68–77. URL: http://dl.acm.org/citation.cfm?id=2227115.2227123.

[30] M. D'Angelo, S. Gerasimou, S. Ghahremani, J. Grohmann, I. Nunes, E. Pournaras, S. Tomforde, On learning in collective self-adaptive systems: State of practice and a 3d framework, in: 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2019, pp. 13–24. doi:10.1109/SEAMS.2019.00012.

[31] A. Pandey, I. Ruchkin, B. Schmerl, D. Garlan, Hybrid planning using learning and model checking for autonomous systems, in: 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2020, pp. 55–64. doi:10.1109/ACSOS49614.2020.00026.

[32] G. J. Lee, J. A. B. Fortes, Improving data-analytics performance via autonomic control of concurrency and resource units, ACM Trans. Auton. Adapt. Syst. 13 (2019). URL: https://doi.org/10.1145/3309539. doi:10.1145/3309539.

[33] R. D. Caldas, A. Rodrigues, E. B. Gil, G. N. Rodrigues, T. Vogel, P. Pelliccione, A hybrid approach combining control theory and ai for engineering self-adaptive systems, in: Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 9–19. URL: https://doi.org/10.1145/3387939.3391595. doi:10.1145/3387939.3391595.

[34] M. Bouton, K. D. Julian, A. Nakhaei, K. Fujimura, M. J. Kochenderfer, Decomposition methods with deep corrections for reinforcement learning, Autonomous Agents and Multi-Agent Systems 33 (2019) 330–352. doi:https://doi.org/10.1007/s10458-019-09407-z.

[35] M. Mikic-Rakic, N. Medvidovic, Support for disconnected operation via architectural self-reconfiguration, in: International Conference on Autonomic Computing, 2004. Proceedings., 2004, pp. 114–121. doi:10.1109/ICAC.2004.1301354.

[36] J. Hao, H.-F. Leung, Achieving socially optimal outcomes in multiagent systems with reinforcement social learning, ACM Trans. Auton. Adapt. Syst. 8 (2013) 15:1–15:23. URL: http://doi.acm.org/10.1145/2517329. doi:10.1145/2517329.

[37] E. Incerto, M. Tribastone, C. Trubiani, Software performance self-adaptation through efficient model predictive control, in: Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, IEEE Press, Piscataway, NJ, USA, 2017, pp. 485–496. URL: http://dl.acm.org/citation.cfm?id=3155562.3155624.

[38] A. Chmielowiec, M. v. Steen, Optimal decentralized formation of k-member partnerships, in: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2010, pp. 154–163. doi:10.1109/SASO.2010.14.

[39] Y. He, Z. Ye, Q. Fu, S. Elnikety, Budget-based control for interactive services with adaptive execution, in: Proc. ICAC, 2012, pp. 105–114.

[40] F. Javed, N. Arshad, Adopt: An adaptive optimization framework for large-scale power distribution systems, in: 2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2009, pp. 254–264. doi:10.1109/SASO.2009.26.

[41] K. Verbeeck, A. Nowé, J. Parent, K. Tuyls, Exploring selfish reinforcement learning in repeated games with stochastic rewards, Autonomous Agents and Multi-Agent Systems 14 (2007) 239–269. URL: http://dx.doi.org/10.1007/s10458-006-9007-0. doi:10.1007/s10458-006-9007-0.

[42] T. Chen, K. Li, R. Bahsoon, X. Yao, Femosaa: Feature-guided and knee-driven multi-objective optimization for self-adaptive software, ACM Trans. Softw. Eng. Methodol. 27 (2018) 5:1–5:50. URL: http://doi.acm.org/10.1145/3204459. doi:10.1145/3204459.

[43] S. Shevtsov, D. Weyns, Keep it simplex: Satisfying multiple goals with guarantees in control-based self-adaptive systems, in: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, ACM, 2016, pp. 229–241.

[44] E. Pournaras, P. Pilgerstorfer, T. Asikis, Decentralized collective learning for self-managed sharing economies, ACM Trans. Auton. Adapt. Syst. 13 (2018) 10:1–10:33. URL: http://doi.acm.org/10.1145/3277668. doi:10.1145/3277668.

[45] B. Escoffier, L. Gourvès, J. Monnot, Fair solutions for some multiagent optimization problems, Autonomous Agents and Multi-Agent Systems 26 (2013) 184–201. URL: http://dx.doi.org/10.1007/s10458-011-9188-z. doi:10.1007/s10458-011-9188-z.

[46] V. Cardellini, E. Casalicchio, V. Grassi, F. Lo Presti, R. Mirandola, Qos-driven runtime adaptation of service oriented architectures, in: Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, ESEC/FSE '09, ACM, New York, NY, USA, 2009, pp. 131–140. URL: http://doi.acm.org/10.1145/1595696.1595718. doi:10.1145/1595696.1595718.

[47] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, R. Mirandola, Moses: A framework for qos driven runtime adaptation of service-oriented systems, IEEE Transactions on Software Engineering 38 (2012) 1138–1159. doi:10.1109/TSE.2011.68.

[48] J. Brownlee, Clever Algorithms, 2011. URL: http://www.cleveralgorithms.com.

[49] C. J. C. H. Watkins, Learning from Delayed Rewards, Ph.D. thesis, King's College, Oxford, 1989.

[50] L. Panait, S. Luke, Cooperative multi-agent learning: The state of the art, Autonomous Agents and Multi-Agent Systems 11 (2005) 387–434. URL: http://dx.doi.org/10.1007/s10458-005-2631-2. doi:10.1007/s10458-005-2631-2.

[51] L. Kraemer, B. Banerjee, Reinforcement learning of informed initial policies for decentralized planning, ACM Trans. Auton. Adapt. Syst. 9 (2014) 18:1–18:32. URL: http://doi.acm.org/10.1145/2668130. doi:10.1145/2668130.

[52] S. Abdallah, V. Lesser, Multiagent reinforcement learning and self-organization in a network of agents, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07, ACM, New York, NY, USA, 2007, pp. 39:1–39:8. URL: http://doi.acm.org/10.1145/1329125.1329172. doi:10.1145/1329125.1329172.

[53] M. Ghavamzadeh, S. Mahadevan, R. Makar, Hierarchical multi-agent reinforcement learning, Autonomous Agents and Multi-Agent Systems 13 (2006) 197–229. URL: https://doi.org/10.1007/s10458-006-7035-4. doi:10.1007/s10458-006-7035-4.

[54] G. Chalkiadakis, C. Boutilier, Bayesian reinforcement learning for coalition formation under uncertainty, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3, AAMAS '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 1090–1097. URL: http://dx.doi.org/10.1109/AAMAS.2004.74. doi:10.1109/AAMAS.2004.74.

[55] G. Chalkiadakis, C. Boutilier, Sequentially optimal repeated coalition formation under uncertainty, Autonomous Agents and Multi-Agent Systems 24 (2012) 441–484. URL: http://dx.doi.org/10.1007/s10458-010-9157-y. doi:10.1007/s10458-010-9157-y.

[56] H. Wang, X. Chen, Q. Wu, Q. Yu, X. Hu, Z. Zheng, A. Bouguettaya, Integrating reinforcement learning with multi-agent techniques for adaptive service composition, ACM Trans. Auton. Adapt. Syst. 12 (2017) 8:1–8:42. URL: http://doi.acm.org/10.1145/3058592. doi:10.1145/3058592.

[57] A. Marinescu, I. Dusparic, S. Clarke, Prediction-based multi-agent reinforcement learning in inherently non-stationary environments, ACM Trans. Auton. Adapt. Syst. 12 (2017) 9:1–9:23. URL: http://doi.acm.org/10.1145/3070861. doi:10.1145/3070861.

[58] B. Torabi, R. Z. Wenkstern, R. Saylor, A collaborative agent-based traffic signal system for highly dynamic traffic conditions, Autonomous Agents and Multi-Agent Systems 34 (2020). doi:10.1007/s10458-019-09434-w.

[59] J. Wildstrom, P. Stone, E. Witchel, R. J. Mooney, M. Dahlin, Towards self-configuring hardware for distributed computer systems, in: Second International Conference on Autonomic Computing (ICAC'05), 2005, pp. 241–249. doi:10.1109/ICAC.2005.63.

[60] H. Kang, H. Chen, G. Jiang, Peerwatch: A fault detection and diagnosis tool for virtualized consolidation systems, in: Proceedings of the 7th International Conference on Autonomic Computing, ICAC '10, ACM, New York, NY, USA, 2010, pp. 119–128. URL: http://doi.acm.org/10.1145/1809049.1809070. doi:10.1145/1809049.1809070.

[61] D. Kim, S. Park, Reinforcement learning-based dynamic adaptation planning method for architecture-based self-managed software, in: 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2009, pp. 76–85. doi:10.1109/SEAMS.2009.5069076.

[62] G. Chen, Z. Yang, H. He, K. Goh, Coordinating multiple agents via reinforcement learning, Autonomous Agents and Multi-Agent Systems 10 (2005) 273–328. doi:10.1007/s10458-004-4344-3.

[63] D. Kramer, W. Karl, Realizing a proactive, self-optimizing system behavior within adaptive, heterogeneous many-core architectures, in: 2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems, 2012, pp. 39–48. doi:10.1109/SASO.2012.26.

[64] J. Zeppenfeld, A. Herkersdorf, Applying autonomic principles for workload management in multi-core systems on chip, in: Proceedings of the 8th ACM international conference on Autonomic computing, ACM, 2011, pp. 3–10.

[65] T. Zhao, W. Zhang, H. Zhao, Z. Jin, A reinforcement learning-based framework for the generation and evolution of adaptation rules, in: 2017 IEEE International Conference on Autonomic Computing (ICAC), 2017, pp. 103–112. doi:10.1109/ICAC.2017.47.

[66] I. Dusparic, V. Cahill, Distributed w-learning: Multi-policy optimization in self-organizing systems, in: 2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2009, pp. 20–29.

[67] J. Rao, X. Bu, C.-Z. Xu, L. Wang, G. Yin, Vconf: A reinforcement learning approach to virtual machines auto-configuration, in: Proceedings of the 6th International Conference on Autonomic Computing, ICAC '09, ACM, 2009, pp. 137–146.

[68] G. Tesauro, N. K. Jong, R. Das, M. N. Bennani, A hybrid reinforcement learning approach to autonomic resource allocation, in: 2006 IEEE International Conference on Autonomic Computing, 2006, pp. 65–73. doi:10.1109/ICAC.2006.1662383.

[69] J. E. Godoy, I. Karamouzas, S. J. Guy, M. Gini, Adaptive learning for multi-agent navigation, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015, pp. 1577–1585. URL: http://dl.acm.org/citation.cfm?id=2772879.2773353.

[70] W. T. L. Teacy, G. Chalkiadakis, A. Farinelli, A. Rogers, N. R. Jennings, S. McClean, G. Parr, Decentralized bayesian reinforcement learning for online agent collaboration, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS '12, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2012, pp. 417–424. URL: http://dl.acm.org/citation.cfm?id=2343576.2343636.

[71] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1992.

[72] V. Nallur, N. Cardozo, S. Clarke, Clonal plasticity: A method for decentralized adaptation in multi-agent systems, in: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '16, ACM, New York, NY, USA, 2016, pp. 122–128. URL: http://doi.acm.org/10.1145/2897053.2897067. doi:10.1145/2897053.2897067.

[73] L. Konig, H. Schmeck, A completely evolvable genotype-phenotype mapping for evolutionary robotics, in: 2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2009, pp. 175–185. doi:10.1109/SASO.2009.20.

[74] M. A. M. d. Oca, T. Stuetzle, M. Birattari, M. Dorigo, Incremental social learning applied to a decentralized decision-making mechanism: Collective learning made faster, in: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2010, pp. 243–252. doi:10.1109/SASO.2010.28.

[75] M. Shaukat, M. Chitre, Bio-inspired practicalities: Collective behaviour using passive neighbourhood sensing, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015, pp. 267–277. URL: http://dl.acm.org/citation.cfm?id=2772879.2772916.

[76] B. An, V. Lesser, D. Westbrook, M. Zink, Agent-mediated multi-step optimization for resource allocation in distributed sensor networks, in: The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '11, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2011, pp. 609–616. URL: http://dl.acm.org/citation.cfm?id=2031678.2031704.

[77] C. Lee, J. Suzuki, An immunologically-inspired autonomic framework for self-organizing and evolvable network applications, ACM Trans. Auton. Adapt. Syst. 4 (2009) 22:1–22:34. URL: http://doi.acm.org/10.1145/1636665.1636668. doi:10.1145/1636665.1636668.

[78] A. J. Ramirez, B. H. C. Cheng, P. K. McKinley, An evolutionary approach to network self-organization and resilient data diffusion, in: 2011 IEEE Fifth International Conference on Self-Adaptive and Self-Organizing Systems, 2011, pp. 198–207. doi:10.1109/SASO.2011.31.

[79] A. J. Ramirez, D. B. Knoester, B. H. C. Cheng, P. K. McKinley, Applying Genetic Algorithms to Decision Making in Autonomic Computing Systems, in: Proc. ICAC, ACM, 2009, pp. 97–106.

[80] A. J. Ramirez, B. H. C. Cheng, P. K. McKinley, B. E. Beckmann, Automatically Generating Adaptive Logic to Balance Non-functional Tradeoffs During Reconfiguration, in: Proc. ICAC, ACM, 2010, pp. 225–234.

[81] G. G. Pascual, M. Pinto, L. Fuentes, Run-time adaptation of mobile applications using genetic algorithms, in: 2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013, pp. 73–82. doi:10.1109/SEAMS.2013.6595494.

[82] Z. Coker, D. Garlan, C. Le Goues, SASS: Self-adaptation using stochastic search, in: Proc. SEAMS, IEEE, 2015, pp. 168–174.

[83] T. Gabor, L. Belzner, T. Phan, K. Schmid, Preparing for the unexpected: Diversity improves planning resilience in evolutionary algorithms, in: 2018 IEEE International Conference on Autonomic Computing (ICAC), 2018, pp. 131–140. doi:10.1109/ICAC.2018.00023.

[84] C. Kinneer, Z. Coker, J. Wang, D. Garlan, C. L. Goues, 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, SEAMS '18, ACM, New York, NY, USA, 2018, pp. 40–50. URL: http://doi.acm.org/10.1145/3194133.3194145. doi:10.1145/3194133.3194145.

[85] M. A. Langford, G. A. Simon, P. K. McKinley, B. H. C. Cheng, Applying evolution and novelty search to enhance the resilience of autonomous systems, in: Proceedings of the IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '19, IEEE Press, 2019, p. 63–69. URL: https://doi.org/10.1109/SEAMS.2019.00017. doi:10.1109/SEAMS.2019.00017.

[86] E. M. Fredericks, B. DeVries, B. H. C. Cheng, Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty, in: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, ACM, New York, NY, USA, 2014, pp. 17–26. URL: http://doi.acm.org/10.1145/2593929.2593937. doi:10.1145/2593929.2593937.

[87] I. Habib, A. Anjum, R. Mcclatchey, O. Rana, Adapting scientific workflow structures using multi-objective optimization strategies, ACM Trans. Auton. Adapt. Syst. 8 (2013) 4:1–4:21.

[88] E. M. Fredericks, Automatically hardening a self-adaptive system against uncertainty, in: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '16, ACM, New York, NY, USA, 2016, pp. 16–27. URL: http://doi.acm.org/10.1145/2897053.2897059. doi:10.1145/2897053.2897059.

[89] C. Kinneer, R. v. Tonder, D. Garlan, C. L. Goues, Building reusable repertoires for stochastic self-* planners, in: 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2020, pp. 222–231. doi:10.1109/ACSOS49614.2020.00045.

[90] S. S. Andrade, J. de A. Macêdo, A Search-Based Approach for Architectural Design of Feedback Control Concerns in Self-Adaptive Systems, in: Proc. SASO, IEEE, 2013, pp. 61–70.

[91] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, P. Plebani, Paws: A framework for executing adaptive web-service processes, IEEE Software 24 (2007) 39–46. doi:10.1109/MS.2007.174.

[92] A. Elkhodary, N. Esfahani, S. Malek, Fusion: A framework for engineering self-tuning self-adaptive software systems, in: Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE '10, ACM, New York, NY, USA, 2010, pp. 7–16. URL: http://doi.acm.org/10.1145/1882291.1882296. doi:10.1145/1882291.1882296.

[93] J. Wu, E. H. Durfee, Automated resource-driven mission phasing techniques for constrained agents, in: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '05, ACM, New York, NY, USA, 2005, pp. 331–338. URL: http://doi.acm.org/10.1145/1082473.1082524. doi:10.1145/1082473.1082524.

[94] J. Wu, E. H. Durfee, Sequential resource allocation in multiagent systems with uncertainties, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07, ACM, New York, NY, USA, 2007, pp. 114:1–114:8. URL: http://doi.acm.org/10.1145/1329125.1329265. doi:10.1145/1329125.1329265.

[95] E. Feo Flushing, L. M. Gambardella, G. A. Di Caro, On decentralized coordination for spatial task allocation and scheduling in heterogeneous teams, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, AAMAS '16, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2016, pp. 988–996. URL: http://dl.acm.org/citation.cfm?id=2937029.2937070.

[96] D. A. Dolgov, M. R. James, M. E. Samples, Combinatorial resource scheduling for multiagent mdps, in: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07, ACM, New York, NY, USA, 2007, pp. 201:1–201:8. URL: http://doi.acm.org/10.1145/1329125.1329369. doi:10.1145/1329125.1329369.

[97] B. Golden, P. Perny, Infinite order lorenz dominance for fair multiagent optimization, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1, AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2010, pp. 383–390. URL: http://dl.acm.org/citation.cfm?id=1838206.1838260.

[98] A. Pandey, G. A. Moreno, J. Cámara, D. Garlan, Hybrid planning for decision making in self-adaptive systems, in: 2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2016, pp. 130–139. doi:10.1109/SASO.2016.19.

[99] G. A. Moreno, J. Cámara, D. Garlan, B. Schmerl, Efficient decision-making under uncertainty for proactive self-adaptation, in: 2016 IEEE International Conference on Autonomic Computing (ICAC), 2016, pp. 147–156. doi:10.1109/ICAC.2016.59.

[100] S. Iannucci, S. Abdelwahed, A probabilistic approach to autonomic security management, in: 2016 IEEE International Conference on Autonomic Computing (ICAC), 2016, pp. 157–166. doi:10.1109/ICAC.2016.12.

[101] A. K. Ramakrishnan, N. Z. Naqvi, Z. W. Bhatti, D. Preuveneers, Y. Berbers, Learning deployment trade-offs for self-optimization of internet of things applications, in: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), USENIX, San Jose, CA, 2013, pp. 213–224. URL: https://www.usenix.org/conference/icac13/technical-sessions/presentation/ramakrishnan.

[102] M. Khan, D. Turgut, L. Bölöni, Optimizing coalition formation for tasks with dynamically evolving rewards and nondeterministic action effects, Autonomous Agents and Multi-Agent Systems 22 (2011) 415–438. doi:10.1007/s10458-010-9134-5.

[103] A. Angelidakis, G. Chalkiadakis, Factored mdps for optimal prosumer decision-making, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015, pp. 503–511. URL: http://dl.acm.org/citation.cfm?id=2772879.2772944.

[104] C. Basich, J. Svegliato, K. H. Wray, S. Witwicki, J. Biswas, S. Zilberstein, Learning to optimize autonomy in competence-aware systems, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '20, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2020, p. 123–131. URL: https://dl.acm.org/doi/10.5555/3398761.3398781.

[105] M. Scheerer, M. Rapp, R. Reussner, Design-time validation of runtime reconfiguration strategies: An environmental-driven approach, in: 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2020, pp. 75–81. doi:10.1109/ACSOS49614.2020.00028.

[106] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proceedings of the American Mathematical

society 7 (1956) 48–50.
[107] R. C. Prim, Shortest connection networks and some generalizations, The Bell System Technical Journal 36 (1957) 1389–1401. doi:10.1002/j.1538-7305.1957.tb01515.x.
[108] T. C. Service, J. A. Adams, Coalition formation for task allocation: Theory and algorithms, Autonomous Agents and Multi-Agent Systems 22 (2011) 225–248. URL: http://dx.doi.org/10.1007/s10458-010-9123-8. doi:10.1007/s10458-010-9123-8.
[109] M. M. Weerdt, Y. Zhang, T. Klos, Multiagent task allocation in social networks, Autonomous Agents and Multi-Agent Systems 25 (2012) 46–86. URL: http://dx.doi.org/10.1007/s10458-011-9168-3. doi:10.1007/s10458-011-9168-3.
[110] S. Bhola, M. Astley, R. Saccone, M. Ward, Utility-aware resource allocation in an event processing system, in: 2006 IEEE International Conference on Autonomic Computing, 2006, pp. 55–64. doi:10.1109/ICAC.2006.1662382.
[111] B. Seshasayee, R. Nathuji, K. Schwan, Energy-aware mobile service overlays: Cooperative dynamic power management in distributed mobile systems, in: Fourth International Conference on Autonomic Computing (ICAC'07), 2007, pp. 6–6. doi:10.1109/ICAC.2007.14.
[112] Y. Ying, R. Birke, C. Wang, L. Y. Chen, N. Gautam, Optimizing energy, locality and priority in a mapreduce cluster, in: Autonomic Computing (ICAC), 2015 IEEE International Conference on, IEEE, 2015, pp. 21–30.
[113] S. Ghanbari, G. Soundararajan, J. Chen, C. Amza, Adaptive learning of metric correlations for temperature-aware database provisioning, in: Fourth International Conference on Autonomic Computing (ICAC'07), 2007, pp. 26–26. doi:10.1109/ICAC.2007.3.
[114] S. Fritsch, A. Senart, D. C. Schmidt, S. Clarke, Scheduling time-bounded dynamic software adaptation, in: Proceedings of the 2008 International Workshop on Software Engineering for Adaptive and Self-managing Systems, SEAMS '08, ACM, New York, NY, USA, 2008, pp. 89–96. URL: http://doi.acm.org/10.1145/1370018.1370035. doi:10.1145/1370018.1370035.
[115] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, M. Trubian, Resource management in the autonomic service-oriented architecture, in: 2006 IEEE International Conference on Autonomic Computing, 2006, pp. 84–92. doi:10.1109/ICAC.2006.1662385.
[116] D. Kusic, N. Kandasamy, Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems, in: 2006 IEEE International Conference on Autonomic Computing, 2006, pp. 74–83. doi:10.1109/ICAC.2006.1662384.
[117] A. Netzer, A. Meisels, R. Zivan, Distributed envy minimization for resource allocation, Autonomous Agents and Multi-Agent Systems 30 (2016) 364–402. URL: http://dx.doi.org/10.1007/s10458-015-9291-7. doi:10.1007/s10458-015-9291-7.
[118] E. Benazera, Planning in stochastic domains for multiple agents with individual continuous resource state-spaces, Autonomous Agents and Multi-Agent Systems 23 (2011) 71–113. doi:10.1007/s10458-010-9131-8.
[119] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney, N. R. Jennings, A distributed algorithm for anytime coalition structure generation, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1, AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2010, pp. 1007–1014. URL: http://dl.acm.org/citation.cfm?id=1838206.1838342.
[120] X. Zhang, V. Lesser, S. Abdallah, Efficient management of multi-linked negotiation based on a formalized model, Autonomous Agents and Multi-Agent Systems 10 (2005) 165–205. URL: http://dx.doi.org/10.1007/s10458-004-6978-6. doi:10.1007/s10458-004-6978-6.
[121] Y. He, Z. Ye, Q. Fu, S. Elnikety, Budget-based control for interactive services with adaptive execution, in: Proceedings of the 9th International Conference on Autonomic computing, ACM, 2012, pp. 105–114.
[122] D. Cooray, E. Kouroshfar, S. Malek, R. Roshandel, Proactive self-adaptation for improving the reliability of mission-critical, embedded, and mobile software, IEEE Transactions on Software Engineering 39 (2013) 1714–1735. doi:10.1109/TSE.2013.36.
[123] N. Bencomo, A. Belaggoun, V. Issarny, Dynamic decision networks for decision-making in self-adaptive systems: A case study, in: 2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013, pp. 113–122. doi:10.1109/SEAMS.2013.6595498.
[124] J. Zhang, R. J. Figueiredo, Autonomic feature selection for application classification, in: 2006 IEEE International Conference on Autonomic Computing, 2006, pp. 43–52. doi:10.1109/ICAC.2006.1662380.
[125] I. Gerostathopoulos, C. Prehofer, T. Bures, Adapting a system with noisy outputs with statistical guarantees, in: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '18, ACM, New York, NY, USA, 2018, pp. 58–68. URL: http://doi.acm.org/10.1145/3194133.3194152. doi:10.1145/3194133.3194152.
[126] J. C. Leite, D. M. Kusic, D. Mossé, L. Bertini, Stochastic approximation control of power and tardiness in a three-tier web-hosting cluster, in: Proceedings of the 7th International Conference on Autonomic Computing, ICAC '10, ACM, New York, NY, USA, 2010, pp. 41–50. URL: http://doi.acm.org/10.1145/1809049.1809056. doi:10.1145/1809049.1809056.
[127] J. Haensel, C. M. Adriano, J. Dyck, H. Giese, Collective risk minimization via a bayesian model for statistical software testing, in: Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 45–56. URL: https://doi.org/10.1145/3387939.3388616. doi:10.1145/3387939.3388616.
[128] N. Esfahani, E. Kouroshfar, S. Malek, Taming uncertainty in self-adaptive software, in: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE '11, ACM, New York, NY, USA, 2011, pp. 234–244. URL: http://doi.acm.org/10.1145/2025113.2025147. doi:10.1145/2025113.2025147.
[129] J. Palmerino, Q. Yu, T. Desell, D. Krutz, Improving the decision-making process of self-adaptive systems by accounting for tactic volatility, in: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 949–961. doi:10.1109/ASE.2019.00092.
[130] P. Zoghi, M. Shtern, M. Litoiu, H. Ghanbari, Designing Adaptive Applications Deployed on Cloud Environments, ACM Trans. Auton. Adapt. Syst. 10 (2016) 1–26.
[131] S. Shevtsov, D. Weyns, M. Maggio, Simca*: A control-theoretic approach to handle uncertainty in self-adaptive systems with guarantees, ACM Trans. Auton. Adapt. Syst. 13 (2019). URL: https://doi.org/10.1145/3328730. doi:10.1145/3328730.
[132] J. Z. Li, J. Chinneck, M. Woodside, M. Litoiu, Fast scalable optimization to configure service systems having cost and quality of service constraints, in: Proceedings of the 6th International Conference on Autonomic Computing, ICAC '09, ACM, 2009, pp. 159–168. URL: http://doi.acm.org/10.1145/1555228.1555268. doi:10.1145/1555228.1555268.

[133] K. Zhang, E. G. Collins, Jr., D. Shi, Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction, ACM Trans. Auton. Adapt. Syst. 7 (2012) 21:1–21:22. URL: http://doi.acm.org/10.1145/2240166.2240171. doi:10.1145/2240166.2240171.
[134] D. Menasce, H. Gomaa, s. Malek, J. Sousa, Sassy: A framework for self-architecting service-oriented systems, IEEE Software 28 (2011) 78–85. doi:10.1109/MS.2011.22.
[135] L. Bao, X. Liu, Z. Xu, B. Fang, Autoconfig: Automatic configuration tuning for distributed message systems, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, ACM, New York, NY, USA, 2018, pp. 29–40. URL: http://doi.acm.org/10.1145/3238147.3238175. doi:10.1145/3238147.3238175.
[136] M. Ennigrou, K. Ghédira, New local diversification techniques for flexible job shop scheduling problem with a multi-agent approach, Autonomous Agents and Multi-Agent Systems 17 (2008) 270–287. doi:10.1007/s10458-008-9031-3.
[137] Y.-X. Wang, Q.-L. Xiang, Z.-D. Zhao, Particle swarm optimizer with adaptive tabu and mutation: A unified framework for efficient mutation operators, ACM Trans. Auton. Adapt. Syst. 5 (2010) 1:1–1:27. URL: http://doi.acm.org/10.1145/1671948.1671949. doi:10.1145/1671948.1671949.
[138] P. Lama, X. Zhou, Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud, in: Proceedings of the 9th international conference on Autonomic computing, ACM, 2012, pp. 63–72.
[139] G. Jung, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, C. Pu, Generating adaptation policies for multi-tier applications in consolidated server environments, in: 2008 International Conference on Autonomic Computing, 2008, pp. 23–32. doi:10.1109/ICAC.2008.21.
[140] N. Kandasamy, S. Abdelwahed, J. P. Hayes, Self-optimization in computer systems via on-line control: application to power management, in: International Conference on Autonomic Computing, 2004. Proceedings., 2004, pp. 54–61. doi:10.1109/ICAC.2004.1301347.
[141] A. Gounaris, C. Yfoulis, R. Sakellariou, M. D. Dikaiakos, A control theoretical approach to self-optimizing block transfer in web service grids, ACM Trans. Auton. Adapt. Syst. 3 (2008) 6:1–6:30. URL: http://doi.acm.org/10.1145/1352789.1352791. doi:10.1145/1352789.1352791.
[142] S. Chuang, A. T. S. Chan, Dynamic qos adaptation for mobile middleware, IEEE Transactions on Software Engineering 34 (2008) 738–752. doi:10.1109/TSE.2008.44.
[143] M. Wang, N. Kandasamy, A. Guez, M. Kam, Adaptive performance control of computing systems via distributed cooperative control: Application to power management in computing clusters, in: 2006 IEEE International Conference on Autonomic Computing, 2006, pp. 165–174. doi:10.1109/ICAC.2006.1662395.
[144] G. A. Moreno, O. Strichman, S. Chaki, R. Vaisman, Decision-making with cross-entropy for self-adaptation, in: 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2017, pp. 90–101. doi:10.1109/SEAMS.2017.7.
[145] P. Pilgerstorfer, E. Pournaras, Self-adaptive learning in decentralized combinatorial optimization - a design paradigm for sharing economies, in: 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2017, pp. 54–64. doi:10.1109/SEAMS.2017.8.
[146] F. Präntare, F. Heintz, An anytime algorithm for optimal simultaneous coalition structure generation and assignment, Autonomous Agents and Multi-Agent Systems 34 (2020). doi:10.1007/s10458-020-09450-1.
[147] N. Zacheilas, V. Kalogeraki, Chess: Cost-effective scheduling across multiple heterogeneous mapreduce clusters, in: 2016 IEEE International Conference on Autonomic Computing (ICAC), 2016, pp. 65–74. doi:10.1109/ICAC.2016.58.
[148] P. Agrawal, A. Kumar, P. Varakantham, Near-optimal decentralized power supply restoration in smart grids, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015, pp. 1275–1283. URL: http://dl.acm.org/citation.cfm?id=2772879.2773315.
[149] S. Miller, S. D. Ramchurn, A. Rogers, Optimal decentralised dispatch of embedded generation in the smart grid, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS '12, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2012, pp. 281–288. URL: http://dl.acm.org/citation.cfm?id=2343576.2343616.
[150] C. J. v. Leeuwen, K. S. Yildirim, P. Pawelczak, Self adaptive safe provisioning of wireless power using dcops, in: 2017 IEEE 11th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2017, pp. 71–80. doi:10.1109/SASO.2017.16.
[151] P. Lalanda, J. McCann, A. Diaconescu, Autonomic Computing - Principles, Design and Implementation, Springer, 2013.
[152] W. B. T. Mock, Pareto Optimality, Springer Netherlands, Dordrecht, 2011, pp. 808–809. URL: https://doi.org/10.1007/978-1-4020-9160-5_341. doi:10.1007/978-1-4020-9160-5_341.
[153] N. Esfahani, A. Elkhodary, S. Malek, A learning-based framework for engineering feature-oriented self-adaptive software systems, IEEE Transactions on Software Engineering 39 (2013) 1467–1493. doi:10.1109/TSE.2013.37.
[154] V. Khare, X. Yao, K. Deb, Performance scaling of multi-objective evolutionary algorithms, in: Evolutionary Multi-Criterion Optimization, Springer, 2003, pp. 376–390.
[155] C. Krupitzer, J. Otto, F. M. Roth, A. Frommgen, C. Becker, Adding Self-Improvement to an Autonomic Traffic Management System, in: Proc. ICAC, IEEE, 2017, pp. 209–214.
[156] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82. doi:10.1109/4235.585893.
[157] M. Harman, B. F. Jones, Search-based software engineering, Inf. Softw. Technol. 43 (2001) 833–839.
[158] M. Harman, P. McMinn, J. Teixeira De Souza, S. Yoo, Search Based Software Engineering: Techniques, Taxonomy, Tutorial, in: Empir. Softw. Eng. Verif., volume 7007 of *LNCS*, Springer, 2012, pp. 1–59.
[159] M. Harman, S. A. Mansouri, Y. Zhang, Search-Based Software Engineering: Trends, Techniques and Applications, ACM Comput. Surv. 45 (2012) 1–61.
[160] M. Harman, E. Burke, J. A. Clark, X. Yao, Dynamic Adaptive Search Based Software Engineering, in: Proc. ESEM, ACM, 2012, pp. 1–8.
[161] L. Wang, Using Search-Based Software Engineering to Handle the Changes with Uncertainties for Self-Adaptive Systems, in: Proc. ESEC/FSE, ACM, 2017, pp. 1014–1017.

[162] L. Wang, Search-Based Adaptation Planning Framework for Self-Adaptive Systems, in: Proc. ICSE-C, IEEE/ACM, 2017, pp. 465–466.
[163] P. Zoghi, M. Shtern, M. Litoiu, Designing Search Based Adaptive Systems: A Quantitative Approach, in: Proc. SEAMS, ACM, 2014, pp. 7–16.
[164] S. S. Andrade, R. J. de Araújo Macêdo, Do Search-Based Approaches Improve the Design of Self-Adaptive Systems ? A Controlled Experiment, in: Proc. Brazilian Symp. Softw. Eng., IEEE, 2014, pp. 101–110.
[165] F. Macías-Escrivá, R. E. Haber Guerra, R. del Toro, V. Hernández, Self-adaptive systems: A survey of current approaches, research challenges and applications, Expert Systems with Applications 40 (2013) 7267–7279. doi:10.1016/j.eswa.2013.07.033.
[166] T. Wong, M. Wagner, C. Treude, Self-adaptive systems: A systematic literature review across categories and domains, 2021. arXiv:2101.00125.
[167] D. Weyns, Software engineering of self-adaptive systems, in: Handbook of Software Engineering, Springer, 2019, pp. 399–443.
[168] T. Patikirikorala, A. Colman, J. Han, L. Wang, A systematic survey on the design of self-adaptive software systems using control engineering approaches, in: 2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012, pp. 33–42.
[169] S. Shevtsov, M. Berekmeri, D. Weyns, M. Maggio, Control-theoretical software adaptation: A systematic literature review, IEEE Transactions on Software Engineering 44 (2018) 784–810. doi:10.1109/TSE.2017.2704579.
[170] T. R. D. Saputri, S.-W. Lee, The application of machine learning in self-adaptive systems: A systematic literature review, IEEE Access 8 (2020) 205948–205967. doi:10.1109/ACCESS.2020.3036037.
[171] O. Gheibi, D. Weyns, F. Quin, Applying machine learning in self-adaptive systems: A systematic literature review, ACM Trans. Auton. Adapt. Syst. 15 (2021). URL: https://doi.org/10.1145/3469440. doi:10.1145/3469440.
[172] C. Krupitzer, T. Sztyler, J. Edinger, M. Breitbach, H. Stuckenschmidt, C. Becker, Hips do lie! a position-aware mobile fall detection system, in: Proceedings of the International Conference on Pervasive Computing and Communications (PerCom), IEEE, 2018, pp. 95–104.