2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS) | 979-8-3503-3744-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ACSOS58161.2023.00028

Towards the decentralized coordination of multiple self-adaptive systems

Paul-Andrei Dragan, Andreas Metzger, Klaus Pohl
paluno (The Ruhr Institute for Software Technology)
University of Duisburg-Essen, Essen, Germany
paul-andrei.dragan@paluno.uni-due.de, andreas.metzger@paluno.uni-due.de, klaus.pohl@paluno.uni-due.de

***Abstract*—When multiple self-adaptive systems share an environment and goals, they may coordinate their adaptations to avoid conflicts and satisfy their goals. There are two approaches to coordination. (1) Logically centralized, where a supervisor has complete control over the self-adaptive systems. Such an approach is infeasible when the systems have different owners or administrative domains. (2) Logically decentralized, where coordination is achieved through direct interactions. Because the individual systems have control over the information they share, decentralized coordination accommodates multiple administrative domains. However, existing techniques do not account simultaneously for local concerns, e.g., preferences, and shared concerns, e.g., conflicts, which may lead to goals not being achieved as expected. We address this shortcoming by expressing both types of concerns within one constraint optimization problem. Our technique, CoADAPT, introduces two types of constraints: preference constraints, expressing local concerns, and consistency constraints, expressing shared concerns. At runtime, the problem is solved in a decentralized way using distributed constraint optimization algorithms. As a first step in realizing CoADAPT, we focus on the coordination of adaptation planning strategies, traditionally addressed only with centralized techniques. We show the feasibility of CoADAPT in an exemplar from cloud computing and analyze experimentally its scalability.**

***Index Terms*—self-adaptive systems, coordination, distributed constraint optimization, cloud computing**

I. Introduction

A self-adaptive system modifies its structure or behavior during operation to continuously meet its goals and requirements [1]. Conceptually, a self-adaptive system can be structured into two main parts: a managed element (ME), realizing the domain logic, and a managing system (MS), responsible for the adaptation concerns [1], [2].

When multiple self-adaptive systems share an environment and have common goals, they may coordinate their adaptations at runtime to avoid conflicting adaptations and to satisfy their common goals [2]. For example, in cloud computing, multiple self-adaptive applications may be deployed on a shared self-adaptive infrastructure; uncoordinated adaptations, e.g., uncoordinated elasticity, may lead to reduced performance [3].

Coordination refers to joint runtime activities carried out by the self-adaptive systems, at the level of their managing systems, for aligning on adaptation. Such activities can include joint goal management [4], adaptation planning [5]–[10], or adaptation enactment [11], among others.

There are two main approaches to coordination. One approach is logically centralized, where a supervisor has complete knowledge and control of the individual self-adaptive systems, e.g., an *application manager* coordinating the elasticity of microservices [12]. This approach may be infeasible, particularly in cases where the systems have different owners or administrative domains, e.g., large-scale compositions of web services [10], cloud federations [13], or smart grids [14].

The other approach is logically decentralized, where coordination is achieved via direct communication between the self-adaptive systems, e.g., by exchanging information on their monitored local state [10], [15]. Because the individual systems have more control over their adaptations and the information they share, decentralized coordination accommodates multiple administrative domains and may improve privacy. However, existing techniques do not properly handle simultaneously both local concerns, e.g., the preference for a particular adaptation in the case of one self-adaptive system, and shared concerns, e.g., conflicts between adaptations pertaining to different self-adaptive systems.

This limitation is due to at least one of the following reasons. First, existing techniques may not explicitly differentiate between the two types of concerns, leading to the sharing of private information [5], [10], [15]. Second, they may make unrealistic assumptions about coordination, e.g., there are no conflicts between adaptations [7]. Third, these techniques may rely on local decision-making only (in contrast to joint decision-making) which may lead to common goals not being achieved as expected [5], [10]

Our idea to address these shortcomings is to express both local and shared concerns within the same constraint optimization problem. Thus, we propose CoADAPT, a decentralized coordination technique for self-adaptive systems, introducing two types of constraints: preference constraints, expressing local concerns, and consistency constraints, expressing shared concerns. At runtime, the problem is solved in a decentralized way using distributed algorithms implemented by each self-adaptive system. With CoADAPT, the self-adaptive systems engage in joint decision-making, conflicts or other shared concerns are explicitly accounted for, and information about local concerns remains private during coordination.

As a suitable base for CoADAPT we have identified the dis-

Research leading to these results received funding from the EU's Horizon 2020 and Horizon Europe R&I programmes under grant agreements 871525 (FogProtect) and 101070455 (DynaBIC).

979-8-3503-3744-0/23/$31.00 ©2023 IEEE
DOI 10.1109/ACSOS58161.2023.00028

tributed constraint optimization problem (DCOP) formalism. DCOPs [16]–[18] are a formal problem setting from the field of multi-agent systems, where agents coordinate the values assigned to the variables they control so as to optimize a global objective. DCOPs were studied for many applications, e.g., smart grids [19], logistics [20], or cloud computing [21].

The benefits of DCOP for coordinating self-adaptive systems are threefold. First, DCOPs can naturally express decentralized or distributed coordination via the construct of *agents*. Second, DCOPs can express both local and shared concerns using constraints. Third, a number of algorithms exist to solve DCOPs that keep certain constraints private to individual agents [18]. Moreover, these algorithms offer different runtime characteristics, e.g., space and time complexities, accommodating various system design requirements, e.g., different numbers of coordinating self-adaptive systems [18].

As a first step in realizing CoADAPT, we focus in this work on coordinating the assignment of adaptation planning strategies [22], traditionally addressed only with centralized techniques [23]–[26]. In contrast to coordination at the level of adaptation planning [5]–[10], coordinating at the higher level of adaptation planning strategies has the important benefit of better supporting heterogeneity [25], [26], e.g., adaptation logics could be realized by means of different techniques or operate at different timescales. Coordination at such a higher level has various applications, e.g., assigning elasticity policies to individual cloud applications for better resource utilization [12] or orchestrating security policies [27].

To summarize, our contribution in this paper consists of:

- We propose an architecture for the decentralized coordination of adaptation planning strategies.
- We propose a formalization for the coordination of strategies taking into account both local and shared concerns.
- We describe how to map such formalization to a DCOP.
- We describe a process for selecting an algorithm for realizing coordination at runtime.
- We show the feasibility of CoADAPT in the Simdex [28] exemplar and analyze experimentally its scalability.

The rest of the paper is structured as follows: Section II gives a brief background on DCOPs and on the task of coordinating adaptation planning strategies. Section III describes our running example. Section IV describes decentralized coordination with CoADAPT. Section V is dedicated to an experimental evaluation. Finally, Section VI gives an overview of related work. Section VII summarizes our work and discusses potential future directions.

II. Background

A. Distributed constraint optimization problem

A distributed constraint optimization problem (DCOP) [16], [18] is formally defined as a tuple $\langle A, X, D, F, \pi, \mu \rangle$, where:

- $A = \{a_1, \ldots, a_n\}$ is the set of *agents*.
- $X = \{x_1, \ldots, x_n\}$ is the set of *variables*.
- $D = \{D_1, \ldots, D_n\}$ is a set of finite variable *domains*, with $D_i = \{\delta_1, \ldots, \delta_p\}$ being the domain of x_i.

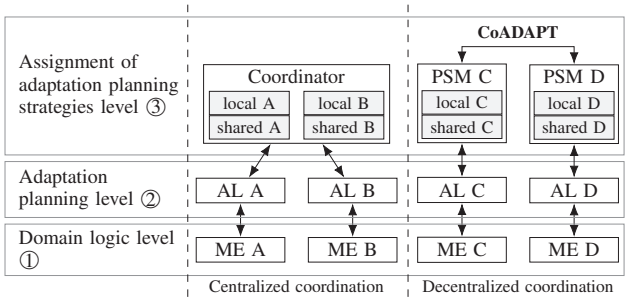

Fig. 1: Coordination of adaptation planning strategies at the conceptual level (ME = managed element, AL = adaptation logic, PSM = planning strategy manager).

- $F = \{f_1, \ldots, f_m\}$ is a set of *constraint functions* (a.k.a. *cost functions* or *constraints*), $f_i : D_l \times \cdots \times D_k \to \mathbb{R}^+ \cup \{\infty\}$. Each f_i has a *scope* $Z_i \subseteq X$. A constraint f_i maps a partial assignment of variables $\delta_{Z_i} \in D_l \times \cdots \times D_k$ from its scope Z_i to a real-valued *cost*. The cardinality of Z_i is called the *arity* of f_i.
- $\pi : X \to A$ maps variables to agents; $\pi(x_i) = a_i$ means that a_i controls how the values from D_i are assigned to x_i, i.e. a_i controls x_i. For simplicity DCOP algorithms assume a one-to-one mapping between agents and variables. We do the same in this paper.
- $\mu(\delta)$ is the *objective function*, where $\delta \in D_1 \times \cdots \times D_n$ is a complete assignment. It is usually assumed that the objective function is just the sum of costs, i.e. $\mu(\delta) = \sum_{f_i \in F} f_i(\delta_{Z_i})$ – a key assumption for many algorithms.

Solving the DCOP means finding a $\delta^\star$ such that:

$$\delta^\star = \underset{\delta \in D_1 \times \cdots \times D_n}{\arg\min}\ \mu(\delta) = \underset{\delta \in D_1 \times \cdots \times D_n}{\arg\min} \sum_{f_i \in F} f_i(\delta_{Z_i}). \quad (1)$$

DCOPs are solved in a distributed way: agents exchange messages with neighbors[1], following certain *algorithms* so as to discover the optimal assignment $\delta^\star$. Many algorithms exist to solve DCOPs; a comprehensive survey is provided in [18].

B. Coordination of adaptation planning strategies

For a self-adaptive system, an adaptation planning strategy is a specification of how adaptation planning, i.e. the creation of sequences of adaptations (a.k.a. adaptation plans), should take place at the level of the adaptation logic [22], [29]. Such strategy could specify, for example, what resources are made available to a self-adaptive system [25], [26], what planning algorithm should be employed [22], or which rule-based adaptation policy should be followed [23], [24].

Coordination at the level of adaptation planning strategies is carried out between multiple self-adaptive systems, where each self-adaptive system may be provided different or managed by a different party, e.g., service-oriented applications [25], [26] or fleets of autonomous vehicles [22]. The individual self-adaptive systems operate, to some extent, independently, i.e. by having local goals and adapting locally, but they abide by

[1]Two agents a_1 and a_2 controlling variables x_1 and x_2, respectively, are *neighbors* if $\exists f \in F$ with scope Z having the property that $x_1, x_2 \in Z$.

an overarching global goal. Moreover, shared concerns may exist between sub-groups of self-adaptive systems with the purpose of exploiting synergies in achieving a shared goal or for avoiding conflicts between adaptations. The adaptation and application logics may be realized differently by each self-adaptive system, thus requiring that coordination is able to handle a large degree of heterogeneity [25], [26].

Conceptually, operations at the level of adaptation planning strategies are carried out at a level above adaptation planning [22]. The result of coordination is an assignment of adaptation planning strategies to each of the self-adaptive systems [25], [26]. Figure 1 shows the three conceptual levels. Level ① is that of the domain or application logic, including the managed elements of all the self-adaptive systems. Level ② is that of adaptation planning, realized by the adaptation logics. Finally, level ③ covers the assignment of adaptation planning strategies to the adaptation logics. Existing works [23]–[26] realize level ③ by means of a central coordinator with complete knowledge of the local and shared concerns. We introduce in Section IV a decentralized alternative that does not make such assumptions.

III. Running example

We illustrate CoADAPT using a cloud system as a running example. Consider two web services part of the same video streaming website: SV1 provides video content and SV2 provides promotional materials, i.e. ads. The two software systems implementing the services share an infrastructure, i.e. they are co-located. The services are affected by uncertainties: the maximum capabilities of the infrastructure, cost of resources, user demand, per-click ad income, "tolerance" of users towards low-quality videos, "willingness" of users to click on ads etc. Therefore, the pre-partitioning of resources at design-time is not feasible and self-adaptation is necessary.

SV1 adapts itself according to two adaptation planning strategies: (A-1), which adjusts video quality within a given amount of available resources, and (A-2), which adjusts the amount of resources to ensure acceptable video quality. Similarly, SV2 has: (B-1), which adjusts ad quality, e.g., images instead of videos, and (B-2), which acquires resources. The common goal of the two services is to maximize revenue.

During low demand the two services prefer (A-2) and (B-2), respectively, to maximize user satisfaction and ad clicks. However, during high demand, there are not enough resources available in the infrastructure to serve both high-quality videos and high-quality ads. If there is no coordination during high demand, the two self-adaptive systems cannot jointly maximize revenue, so they pick their strategies as they believe appropriate based on local state, i.e. (demand, resources, satisfaction) for SV1 and (demand, resources, clicks) for SV2. For example, SV2 tries to optimize ad clicks by acquiring more resources (B-2), while SV1 becomes resources starved so it starts reducing video quality (A-1). This could lead to poor user satisfaction and loss of revenue due to users leaving the system early and not clicking on ads.

IV. Decentralized coordination of adaptation planning strategies with CoADAPT

A. Overview

To coordinate the assignment of adaptation planning strategies between multiple self-adaptive systems we propose a hierarchical architecture for each of the coordinating self-adaptive systems, following the three conceptual levels discussed in Section II-B. Such an architecture enables a clear separation of concerns within a self-adaptive system [2], [30] and allows individual layers to operate at different levels of abstraction and at possibly different timescales [2].

Figure 2 depicts our architecture for two coordinating self-adaptive systems and consists of three main components:

- The managed element (ME) realizes the domain logic of the self-adaptive system and is part of the conceptual level ① (Figure 1).
- The adaptation logic (AL) monitors and adapts the ME. The AL pertains to a particular managing system (MS) and carries out adaptation planning according to an active adaptation planning strategy. The AL is in level ②.
- The planning strategy manager (PSM) monitors the AL and the ME, selects the adaptation planning strategies, and assigns them to the AL. The PSMs are part of level ③ and implement the CoADAPT coordination logic.

Part of the PSM are the local concerns, e.g., preferences for a certain adaptation planning strategy, and shared concerns, e.g., known conflicts between certain adaptation planning strategies of different self-adaptive systems. Both the local and shared concerns influence the strategy assignment process. However, during coordination, the local concerns remain private to a particular system, while the shared concerns are visible to the systems affected by those concerns.

Our architecture is extended to an arbitrary number of self-adaptive systems as follows: 1) each system will have its own set of local concerns, 2) each system will have a set of shared concerns for each coordinating set[2], and 3) direct communication will happen only within each coordinating set.

Realizing the CoADAPT architecture at design-time begins with system designers defining the global, local, and shared goals of the system consisting of multiple self-adaptive systems. The definition of shared goals is done collaboratively between relevant system designers. The system designers also collaborate on prioritizing local and shared goals from the perspective of the global goal of the system. System designers then define possible adaptations at the level of each self-adaptive system and identify potential conflicts or synergies between the adaptations of different self-adaptive systems. The focus of this paper, however, lies on the steps that follow, i.e. the formal specification of coordination and we leave the previously mentioned engineering steps to future work. In the specification steps, the system designers will:

1) Formalize coordination using Section IV-B,
2) Map the formalization to a DCOP using Section IV-C,

[2]A *coordinating set* includes all self-adaptive systems sharing a concern.

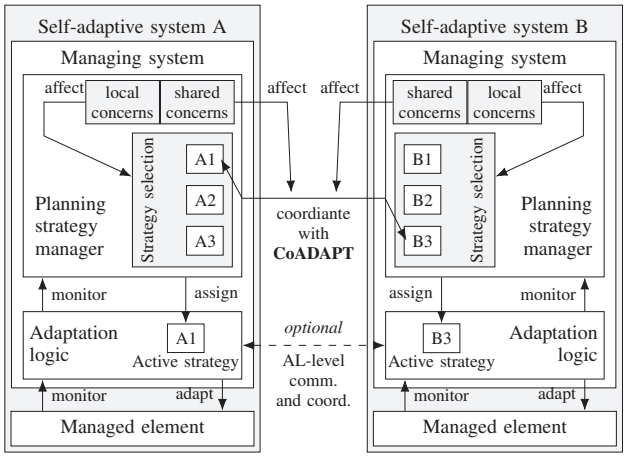

Fig. 2: Architecture for coordinating the assignment of adaptation planning strategies with CoADAPT for a particular instance. Arrows show the flow of information or of commands.

3) Decide on an appropriate decentralized algorithm for solving the DCOP using the reasoning from Section IV-D,
4) Define the update function to update local and shared concerns and reflect them in the DCOP.

Then, at runtime, periodically or event-triggered, e.g., upon major changes, the following should happen autonomously:

1) The update function reflects the current state and goals in the coordination DCOP,
2) The PSMs follow the algorithm to solve the DCOP and obtain the strategy assignments for each ALs,
3) The ALs activate the adaptation planning strategies.

In this paper, we assume the following when realizing the CoADAPT architecture:

- A fixed set of self-adaptive systems and each system has a fixed set of possible strategies.
- Perfect communication during coordination.
- Cooperation among the systems in achieving common goals (in spite of keeping local concerns private).
- (Optional) Further communication between ALs to exchange information relevant for coordination, if necessary (dashed arrow in Figure 2) [31].
- Mechanisms at the level of each PSM to reflect the information from the AL in local and shared concerns, i.e. an *update function*.

B. Formalizing coordination

1) Overview: At runtime, the self-adaptive systems select their adaptation planning strategies so as to best meet common goals, while satisfying local and shared concerns. Therefore, we can formalize the coordinated assignment of adaptation planning strategies as an optimization problem. Given n self-adaptive systems with the architecture from Figure 2, coordination is formalized by the tuple $\langle S, C, O, K, \Sigma, \rho \rangle$:

- $S = S_1 \times \cdots \times S_n$, where S_i is the set of possible states of the ith ME and of its environment, as monitored by the ith AL,
- $C = C_1 \times \cdots \times C_n$, where C_i is the set of possible adaptation planning strategies of the ith AL,
- $O = O_L \times O_S$, where O_S is a set of *shared concerns* and $O_L = O_1 \times \cdots \times O_n$ are *local concerns*,
- $K = K_L \times K_S$, K_S is the set of shared knowledge and $K_L = K_1 \times \cdots \times K_n$ are local knowledge,
- $\Sigma = S \times C \times O \times K$ is the set of complete states of the system composed of the n self-adaptive systems,
- $\rho : \Sigma \times C \to \mathbb{R}^+ \cup \{\infty\}$, where $\rho(\sigma, c')$ gives the cost of the new strategies $c' = (c'_1, c'_2, \ldots, c'_n) \in C$, given $\sigma \in \Sigma$, the current state of the system.

Note that in this paper, *knowledge* means models, i.e. functions $k_i : O_i \times S_i \times C_i \to \mathbb{R}^+ \cup \{\infty\}$, $k_i \in K_i$ and $k_Z : O_S \times S_Z \times C_Z \to \mathbb{R}^+ \cup \{\infty\}$, $k_Z \in K_S$, expressing the *contribution* of strategies from C to local and shared concerns, respectively, given the current state. Z denotes the *scope* of a shared concern, i.e. the coordinating set (see Section IV-A).

The objective of coordination is then to find:

$$c^\star = \arg\min_{c' \in C} \rho(\sigma, c'). \tag{2}$$

It can be said that $\rho(\cdot)$ *predicts* the effect of switching the adaptation planning strategies of all ALs to c' and associates to this effect a cost. Using such predictions to drive adaptation is common for self-adaptive systems [5], [8].

Finally, we emphasize that $c^\star = (c_1^\star, c_2^\star, \ldots, c_n^\star) \in C$ is a globally-optimal solution and does not give guarantees with respect to the optimal satisfaction of local, self-adaptive system-specific goals. This means that one can encounter cases where $c_i^\star$ does not optimally satisfy a local concern O_i.

The *key idea* of CoADAPT is to decompose $\rho(\cdot)$ in a sum of contributions of adaptation planning strategies to local concerns plus a sum of contributions to shared concerns:

$$\rho(\sigma, c') = \sum_{i=1..n} \mathrm{k}_i(o_i, s_i, c'_i) + \sum_{\substack{k_Z \in K_S,\, o_Z \in O_S \\ s_Z \in S_Z}} \mathrm{k}_Z(o_Z, s_Z, c'_Z)\,. \tag{3}$$

This decomposition is general and is suited for systems with both loosely-coupled adaptation planning strategies, i.e. many scopes Z_i, but with cardinality $|Z_i|$ small, and tightly-coupled strategies, i.e. few Z_i, but with big $|Z_i|$. Moreover, this decomposition can express strongly conflicting adaptation planning strategies with $k_Z(\cdot) = \infty$, strongly synergetic strategies with $k_Z(\cdot) = 0$, and any other degree of compatibility with $k_Z(\cdot) \in (0, \infty)$. The same follows for local concerns.

2) Example: We apply the formalization to the running example. A possible design of $\rho(\cdot)$ is:

$$\rho(\sigma, c') = k_1\left(o_1, s_1, c'_1\right) + k_2\left(o_2, s_2, c'_2\right) + k_3\left(o_3, s_1, s_2, c'_1, c'_2\right)\,, \tag{4}$$

where $s_1 \in S_1$, $s_2 \in S_2$ are monitored states; $c'_1 \in C_1$ and $c'_2 \in C_2$ are adaptation planning strategies; $o_1 \in O_1$, $o_2 \in O_2$, $o_3 \in O_S$ are concerns; $k_1 : O_1 \times S_1 \times C_1 \to \mathbb{R}^+$, $k_1 \in K_1$, $k_2 : O_2 \times S_2 \times C_2 \to \mathbb{R}^+$, $k_2 \in K_2$, $k_3 : O_S \times S_1 \times S_2 \times C_1 \times C_2 \to \mathbb{R}^+$, $k_3 \in K_S$ are models. The contribution to local concerns is given by $k_1(\cdot)$ and $k_2(\cdot)$, while the contribution to

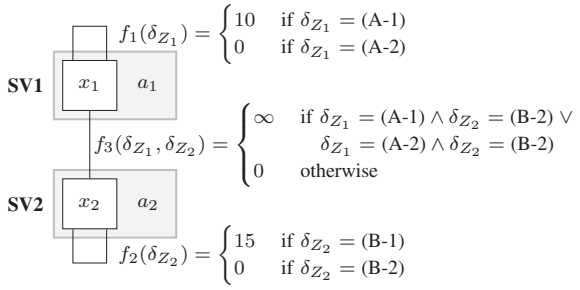

Fig. 3: A mapping of the running example to DCOP.

shared concerns is given by $k_3(\cdot)$; $k_3(\cdot)$ should ensure that (A-1) and (B-2) or (A-2) and (B-2) are not simultaneously selected during high demand, i.e. avoid starving SV1 of resources. The monitored states include the amount and qualities of served videos and ads and the used resources. The local concerns are with respect to the amount and qualities of served videos and ads, respectively, while the shared concern is with respect to the overall revenue generated by the web application.

C. Mapping coordination to DCOP

1) Overview: Next, we introduce a mapping of coordination (Section IV-B) to a DCOP. In our approach, the system integrator specifies, starting from $\langle S, C, O, K, \Sigma, \rho \rangle$, the structure of a DCOP, including variables, agents, and constraints. At runtime, this DCOP is updated dynamically to take into account the new complete state $\sigma \in \Sigma$ of the system (see Section IV-A). Solving the DCOP and obtaining the optimal coordination solution $c^\star$ is done at runtime in a decentralized manner using a distributed constraint optimization algorithm.

2) Variables, domains, and agents: Each $x_i \in X$ corresponds to the adaptation planning strategies selection for self-adaptive system i. The set of possible adaptation planning strategies C_i is represented by the domain D_i of x_i. Agents a_i controlling variables x_i and implementing DCOP algorithms are implemented by the PSM of each self-adaptive system.

3) Mapping concerns to constraints: It is easy to map (3) to DCOP constraints. The contributions to local concerns, i.e. $k_i(\cdot)$, are mapped to DCOP unary constraints, i.e. to f_i with scope cardinality $|S_i| = 1$. We call such constraints *preference constraints* because their main purpose is to specify local preferences towards certain adaptation planning strategies. A strong preference for a certain adaptation planning strategy $\delta_1 \in D_i$ would be specified as $f_i(\delta_1) = 0$. If a strategy $\delta_2 \in D_i$ must be avoided, then $f_i(\delta_2) = \infty$.

We map the contributions of shared concerns, i.e. $k_Z(\cdot)$, to n-ary DCOP constraints, i.e. f_i with $|S_i| > 1$. We call them *consistency constraints* because they quantify *how well* two or more strategies work together. For example, if two adaptation planning strategies, specified as $\delta_1 \in D_j \times D_k$ are in strong conflict, then $f_i(\delta_1) = \infty$. If two strategies, specified as $\delta_2 \in D_j \times D_k$, are strongly synergetic, then $f_i(\delta_2) = 0$.

Separating constraints in preference and consistency constraints gives important privacy benefits. In some algorithms, e.g., DPOP [17], agents exchange information about n-ary constraints with other agents involved in those constraints, while unary constraints stay private.

4) Example: In the case of high demand, a possible mapping of coordination to DCOP is depicted in Figure 3 (variables are white rectangles, agents are gray rectangles, and constraints are lines). The variable x_1 corresponds to the adaptation planning strategy assignment of SV1 and has the domain $D_1 = C_1 = \{\text{(A-1)}, \text{(A-2)}\}$. Variable x_2 corresponds to SV2 and its domain is given by $D_2 = C_2 = \{\text{(B-1)}, \text{(B-2)}\}$.

We map $k_1(\cdot)$ and $k_2(\cdot)$ to the preference constraints f_1 and f_2, respectively, and $k_3(\cdot)$ to the consistency constraint f_3. The constraint f_1 is associated to x_1, while f_2 is associated to x_2. The consistency constraint f_3 high costs to the cases expected to lead to poor revenue (Section IV-B2). The solution of this DCOP would then be $\delta^* = (\text{(A-2)}, \text{(B-1)})$, meaning that SV1 will acquire more resources during high demand, while SV2 will adapt the quality of ads.

D. Selecting a DCOP algorithm

System designers have a significant number of options when choosing a DCOP algorithm for coordination. We base the following guidelines on the theoretical properties of DCOP algorithms and recommendations from [18].

1) Constrained resources: Many self-adaptive systems are constrained by the resources at their disposal, which affects coordination and the choice of DCOP algorithm. In case of limited memory, e.g., the coordination logic runs on edge devices [6], algorithms trading more computation for less memory usage, e.g., the ADOPT search-based algorithm [16], are preferred. In case of intermittent network connectivity, e.g., mobile IoT, it can be better to send fewer messages and inference-based algorithms like DPOP [17] are preferred.

2) Complexity of coordination task: When the behaviors of coordinating self-adaptive systems are tightly coupled, e.g., robotic systems [5] with stringent safety requirements, conflicts between their adaptation planning strategy can lead to cycles in the DCOP constraint graph. Inference-based algorithms, e.g., DPOP, are not suitable for such systems because an increase in cycles can result in exponential increases in coordination overheads, e.g., size of messages or execution time. Instead, search-based algorithms are preferred.

3) Example: Because our example does not have any cycles in its DCOP, the DPOP algorithm would be a good choice as it has been shown to a very good performance in such cases.

V. Experimental evaluation

We aim to answer the following research questions:

(**RQ1**) *What is the impact of CoADAPT in comparison to uncoordinated adaptation?*

(**RQ2**) *How does CoADAPT scale, in terms of communication and execution overheads, with increasing system sizes?*

A. Overall experiment design

1) Self-adaptive exemplar: To answer (**RQ1**) and (**RQ2**), we evaluate CoADAPT[3] in an exemplar of a cloud system consisting of an infrastructure, providing computational resources,

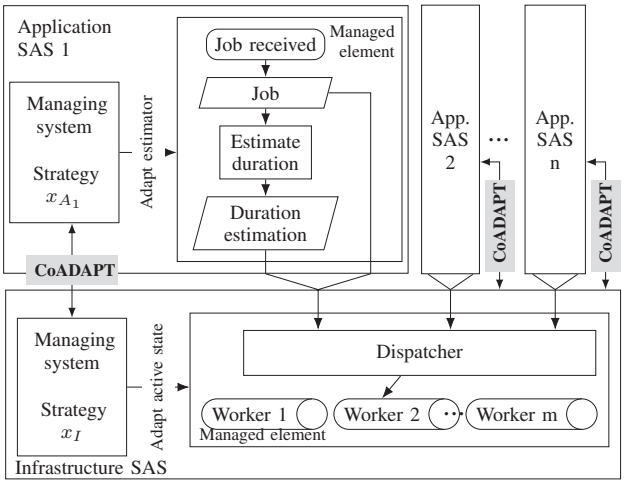

Fig. 4: Job-dispatching self-adaptive system exemplar.

and multiple applications, consuming the resources. The MS of the infrastructure, a.k.a. the infrastructure manager (IM), and those of the applications, a.k.a. the application managers (AMs), coordinate on their adaptation planning strategies with respect to costs and user satisfaction. We decided on this problem setting due to our interest in cloud computing and because similar problems have been well-studied by the community [7], [15], [32], [33].

Concretely, we carry out our experiments in the Simdex exemplar [28]. Simdex simulates a job-dispatching system where a backend distributes jobs between workers. Originally, Simdex is a monolithic self-adaptive system with centralized adaptation logic. To evaluate CoADAPT, we extend Simdex to a setting involving multiple self-adaptive systems: an infrastructure and n applications (Figure 4).

The ME of the infrastructure, consists of workers and a dispatcher. The dispatcher assigns jobs to workers based on the workers' availability, load, and a *job duration estimation*. The IM adapts workers' state (active or inactive) according to two possible adaptation planning strategies: *performance* (I-P), where all workers are active at all times, and *resource consumption minimization* (I-E), where more workers are activated with increasing demand.

Figure 4 includes n applications. The ME of each application is an estimator estimating the duration of each job. The AM adapts the job duration estimator upon having processed new jobs to improve future predictions; adaptation is done according to three possible adaptation planning strategies: *statistics-based* (A-A), using a running average of job durations, *machine learning-based* (A-N), using a backpropagation algorithm, and *simple* (A-S), which does a constant update using predefined duration limits, regardless of the real duration.

To realize the architecture from Figure 4 we reuse the adaptation logics provided by Simdex, consolidating them as adaptation planning strategies. We extended the simulation

[3]Code and results of our experiments are available at `github.com/pauldragan/acsos2023-dec-coord`.

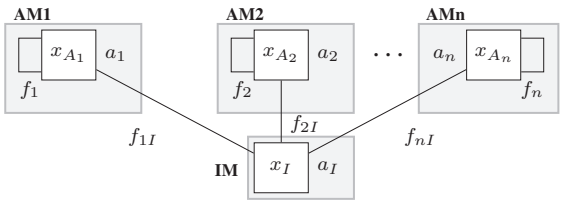

Fig. 5: DCOP specification of coordination in the exemplar.

to trigger a coordination step with period T_R. To assess adaptation effectiveness, we use the two metrics provided by Simdex: *resource consumption*, measuring the number of active workers, and *dissatisfaction*, counting delayed and late jobs[4] (see Sections 4.1 and 4.2 of [28]). Next, we apply CoADAPT to the subject system.

2) Formalizing coordination: First, we formalize coordination by following Section IV-B. We define $\langle S, C, O, K, \Sigma, \rho \rangle$:

- $S = S_I \times S_{A_1} \times \cdots \times S_{A_n}$, where S_I is the set of worker states and S_{A_i} are the sets of states of duration estimators,
- $C = C_I \times C_{A_1} \times \cdots \times C_{A_n}$, where $C_I = \{\text{(I-E)}, \text{(I-P)}\}$ and $C_{A_i} = \{\text{(A-S)}, \text{(A-A)}, \text{(A-N)}\}$,
- $O = O_L \times O_S$, where $O_L = O_{A_1} \times \cdots \times O_{A_n}$, with $O_{A_i} = O_{e_i} \times O_{d_i} \times O_{l_i}$ and O_{e_i} are local concerns related to operational costs, O_{d_i} are for the number of delayed jobs, and O_{l_i} are related to the number of late jobs; the shared concerns $O_S = O_{1I} \times \cdots \times O_{nI}$, with $O_{iI} = O_{e_{iI}} \times O_{d_{iI}} \times O_{l_{iI}}$, are between the applications and the infrastructure and follow a similar logic with respect to costs, delayed jobs, and late jobs; for simplicity, we assume that there are no local concerns at the level of the infrastructure,
- $K = K_L \times K_S$, where $K_L = K_{A_1} \times \cdots \times K_{A_n}$ with $(e_i, d_i, l_i) \in K_{e_i} \times K_{d_i} \times K_{l_i} = K_{A_i}$ giving the contributions of application adaptation planning strategies to local concerns; and $K_S = K_{1I} \times \cdots \times K_{nI}$, with $(e_{iI}, d_{iI}, l_{iI}) \in K_{e_{iI}} \times K_{d_{iI}} \times K_{l_{iI}} = K_{iI}$ giving the contributions to shared concerns.

Then, $\rho : \Sigma \times C \to \mathbb{R}^+ \cup \{\infty\}$ is given by:

$$\rho(\sigma, c') = \sum_{i=1..n} e_i(\sigma_i, c'_i) + d_i(\sigma_i, c'_i) + l_i(\sigma_i, c'_i) + \sum_{i=1..n} e_{iI}(\sigma_{iI}, c'_i) + d_{iI}(\sigma_{iI}, c'_{iI}) + l_{iI}(\sigma_{iI}, c'_{iI}), \quad (5)$$

where $\sigma_i \in O_i \times S_i$, $\sigma_{iI} \in O_i \times S_I \times S_i$, $c_i \in C_i$, $c_{iI} \in C_i \times C_I$.

3) Mapping to a DCOP: Here we follow the steps from Section IV-C. Figure 5 shows a graphical representation of the resulting DCOP. Variables model the adaptation planning strategy selection for each of the AMs, i.e. $x_{A_1}, \ldots, x_{A_n} \in X$, and for the IM, i.e. $x_I \in X$. The domains of x_{A_i} are $D_{A_i} = C_{A_i} = \{\text{(A-S)}, \text{(A-A)}, \text{(A-N)}\}$ and the domain of x_I is $D_I = C_I \{\text{(I-E)}, \text{(I-P)}\}$. We map (5) to DCOP constraints as follows. Given the assignments δ_{A_i} of variables x_{A_i} and δ_I of x_I, the optimization objective is:

[4]*Delayed* jobs are processed in less than 30 seconds, but in two times as long than for a similar reference job. *Late* jobs are processed in more than 30 seconds.

TABLE I: Statistics of the experiments carried out for (**RQ1**). The values are averaged over ten runs.

Experiment	Jobs delayed		Jobs late		Workers	
	avg.	std.	avg.	std.	avg.	std.
Baseline 1	1.41%	0.02%	3.48%	0.12%	1.06	0.00
Baseline 2	0.58%	0.02%	2.77%	0.19%	4.00	0.00
Coordination	0.70%	0.19%	2.87%	0.26%	3.52	0.47

$$\mu(\delta_I, \delta_{A_1}, \dots) = \sum_{i=1..n} f_{iI}(\delta_{A_i}, \delta_I) + \sum_{i=1..n} f_i(\delta_{A_i}) \,. \quad (6)$$

The consistency constraints between each x_{A_i} and x_I are given by $f_{1I}, \dots, f_{nI} \in F$ with:

$$f_{iI}(\delta_{A_i}, \delta_I) = e_{iI}(\delta_{A_i}, \delta_I) + d_{iI}(\delta_{A_i}, \delta_I) + l_{iI}(\delta_{A_i}, \delta_I) \,. \quad (7)$$

The preference constraints $f_1, \dots, f_n \in F$ are given by:

$$f_i(\delta_{A_i}) = e_i(\delta_{A_i}) + d_i(\delta_{A_i}) + l_i(\delta_{A_i}) \,. \quad (8)$$

We reused some of the notation from Section V-A2 for the e_i, d_i, l_i, e_{iI}, d_{iI}, and l_{iI} functions. As they were not our focus, we realized these functions as simple key-value maps, where keys are adaptation planning strategies; we chose the map values experimentally to match the effect of strategies assessment metrics. Important for the experiments is that e_{iI}, d_{iI}, and l_{iI} have a linear structure, e.g., $e(\delta_{A_i}, \delta_I) = e_i(\delta_{A_i}) + e_I(\delta_I)$, where e_I gives a contribution of an infrastructure strategy to energy costs (same logic applies to d_{iI}, and l_{iI}).

4) Algorithm selection: Because the constraint graph of the coordination DCOP is acyclic, we choose the DPOP algorithm [17] due to its good performance and privacy features. We use the implementation of DPOP from the pyDcop library [34].

B. Experiments for answering (*RQ1*)

1) Design: We study the effect of coordination between the AMs and the IM on the subject system under evolving requirements at the level of the n application self-adaptive systems. Such situations are typical of self-adaptive system, where it is expected that requirements will change during operation [35]. To simulate changes in requirements and goals we introduce small random variations in e_i, d_i, l_i, every $T_R = 180$ days, i.e. the coordination period, and update the DCOP accordingly. We choose $T_R = 180$ days so as to reflect that requirements are slow-changing.

We then compare the values of the assessment metrics (Section V-A1) when the AMs and the IM coordinate against when they do not coordinate. This results in three experiments:

Baseline 1. The AMs change their individual adaptation planning strategy assignments in response to new e_i, d_i, and l_i, without coordinating, i.e. they individually optimize their preference constraints. The strategy of the IM is fixed to (I-E).

Baseline 2. Same as the previous experiment, but the adaptation planning strategy of the IM is fixed to (I-P).

Coordination. The self-adaptive systems coordinate on the adaptation planning strategy assignments with CoADAPT.

All experiments include *five* applications and *one* infrastructure. We have decided on these numbers so as to ensure sufficient complexity, while keeping simulation times to reasonable durations. The AMs and IM have access to all adaptation planning strategies from Section V-A1 and the infrastructure has four workers (as in the original Simdex paper [28]).

As our experiments involve randomized elements, i.e. e_i, d_i, and l_i, we run all experiments ten times, each run with a different seed of the random number generator. Then we compute the average of delayed, late jobs, and number of active workers. Of course, the seed is same between the **Baseline 1**, **Baseline 2**, and **Coordination** of the same run.

2) Results: The top row in Figure 6 shows identical plots depicting the workload. Note that this plot is only shown for explaining the peaks in other plots from Figure 6. Workload itself does not drive the assignment of adaptation planning strategies; the assignment is done by minimizing (6).

Row four of Figure 6 shows the value of the objective function change every $T_R = 180$ days due to new requirements. **Baseline 1** exhibits different characteristics than **Baseline 2** because of the different contributions of (I-E) and (I-P). Interestingly, **Coordination** seems to be a mixture of **Baseline 1** and **Baseline 2**, e.g., in the first half of 2017 the objective function has the same value as **Baseline 1**, while in the second half the value is identical to that of **Baseline 2**. Indeed, the objective function of **Coordination** is always the smallest between the those of **Baseline 1** and **Baseline 2**; given that coordination is expressed as a minimization problem, this plot indicates that coordination behaves correctly.

The bottom row of Figure 6 shows the assigned adaptation planning strategies. Each row corresponds to a specific AM or to the IM and cells are the active strategies at a particular point in time. The strategies selected by the AMs being the same in the three experiments is explained by the linear construction of e_{iI}, d_{iI}, and l_{iI}: minimizing the objective function is equivalent to individually minimizing the terms corresponding to the AMs (8) and then coordinating on the optimal IM strategy (7). **Coordination** achieves the lowest objective function values by always selecting the optimal IM adaptation planning strategy between (I-E) and (I-P).

The second and third rows of Figure 6 show the delays in job processing and the average number of active workers, respectively. Although it is difficult to assess the effects of coordination solely from these plots, Table I shows that **Coordination** achieves better results than **Baseline 1** but slightly worse than **Baseline 2** in terms of late and delayed jobs. An explanation is in the third row of Figure 6: **Baseline 1** attempts to minimize the number of active workers (I-E), whereas **Baseline 2** maintains four active workers at all times (I-P). **Coordination** switches between (I-E) and (I-P), resulting in periods where all workers are simultaneously active and periods where workers are active during high demand. Table I shows that, indeed, the average number of active workers in **Coordination** is between **Baseline 1** and **Baseline 2**.

3) Discussion: These results validate CoADAPT: through coordination, the AMs and IM select their adaptation planning strategies so as to jointly minimize the objective function. One could say, based on Figure 6 and Table I, that the results are not strikingly in favor of **Coordination**. We should keep in mind

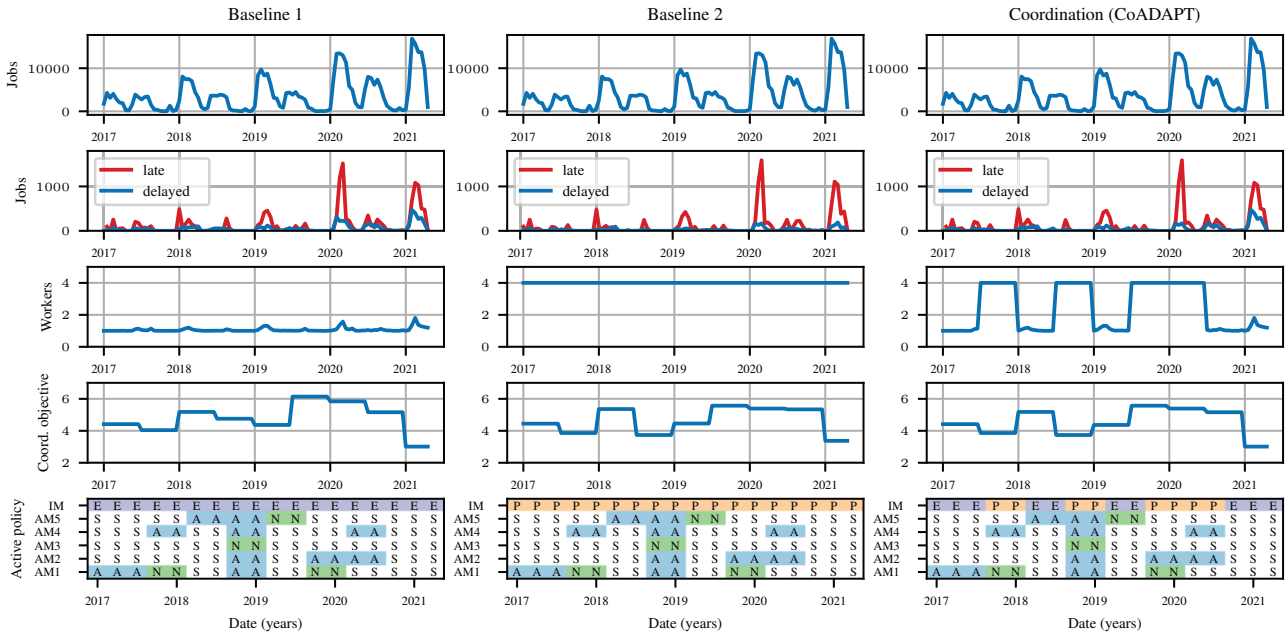

Fig. 6: **Rows (top to bottom)**: **Row 1** Workload evolution expressed as the number of jobs. **Row 2** Number of jobs which have become *delayed* or *late*. **Row 3** Infrastructure utilization expressed as the number of active workers. **Row 4** Evolution of the coordination objective. **Row 5** Adaptation planning strategies over time, for the IM (E is (I-E), P is (I-P)) and the AMs (S is (A-S), A is (A-A), N is (A-N)). **(Columns)** Three experiments (left to right): **Baseline 1**, **Baseline 2**, and **Coordination**.

two aspects: 1) the results achieved by **Baseline 1** and **Baseline 2** are already very good, therefore any improvement will not appear as significant, and 2) coordination is purely driven by an open-loop objective function, constructed experimentally. Devising the best objective function for Simdex was not the focus of this work, but better objective functions could be considered in the future, e.g., by employing performance modeling or machine learning techniques [12].

C. Experiments for answering (*RQ2*)

1) Design: We study the scalability of CoADAPT for increasing numbers of applications and strategies. We carry out this experiment using pyDcop only, and not Simdex, because we are interested in coordination overheads and not in the results of coordination. The experiments run on a system with an Intel i7-1065G7 CPU and 16GB RAM.

2) Results: Figure 7 shows the effect of increasing the number of AMs. For simplicity, we assume $|C_i| = |C_I|$. To see the relation between increasing the number of applications and that of increasing the number of adaptation planning strategies, we ran the experiment for three domain sizes. We observe: 1) the number of exchanged messages increases linearly with the AMs and is not influenced by the domain sizes, 2) the total size of messages increases linearly with the AMs and depends on the domain sizes, i.e. larger domains result in larger messages, and 3) the execution time increases linearly with the AMs and larger domains give longer execution times.

These results are expected given the characteristics of the DPOP algorithm and our simple coordination task (acyclic graph – see Figure 5). In DPOP the number of exchanged messages increases linearly with the number of agents and does not depend on domain sizes. An increase in the number of AMs gives an increase in the number of exchanged messages with the same factor. The total size of messages in DPOP increases exponentially in the worst case and depends on the number of cycles in the problem graph and on domain sizes. However, because we have no cycles, we are actually in a best case scenario, giving a linear increase of total message size. A similar logic follows for time-complexity.

Figure 8 shows the effect of increasing numbers of strategies. The results support previous discussions: the number of exchanged messages does not depend on the domain sizes and the total size of messages increases linearly due to our acyclic graph. The execution time increases quadratically with the domain size. The reason is that D_{A_i} and D_I are increasing simultaneously, resulting in a quadratic increase of the time required for evaluating the consistency constraints f_{iI}.

3) Discussion: CoADAPT shows promising scalability, with coordination overheads increasing mostly linearly or quadratically with increasing problem sizes. CoADAPT could effectively coordinate many applications with an infrastructure for large-scale cloud self-adaptive systems. However, the DPOP algorithm may limit more complex coordination settings, where the DCOP constraint graph has many cycles. In such cases, it would be worthwhile to consider other DCOP algorithms, e.g., those exploiting parallelization [16].

D. Threats to the validity of experiments

We have identified and addressed the following threats to the *internal validity*: 1) the size of the validation scenario, i.e. a limited number of AMs and adaptation planning strategies, which we addressed by including a scalability experiment, 2)

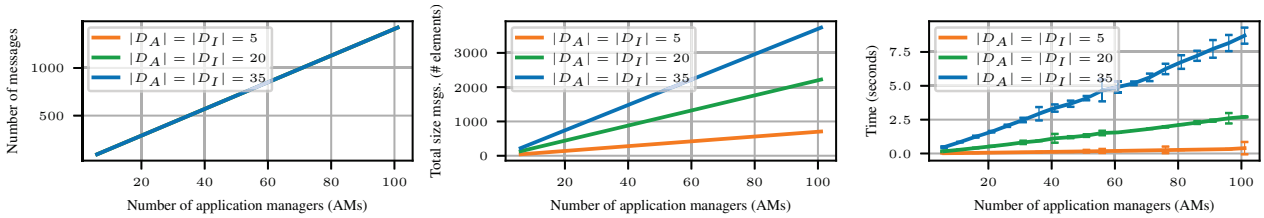

Fig. 7: Effect of increasing the number of AMs on coordination overheads. Error bars show three standard deviations.

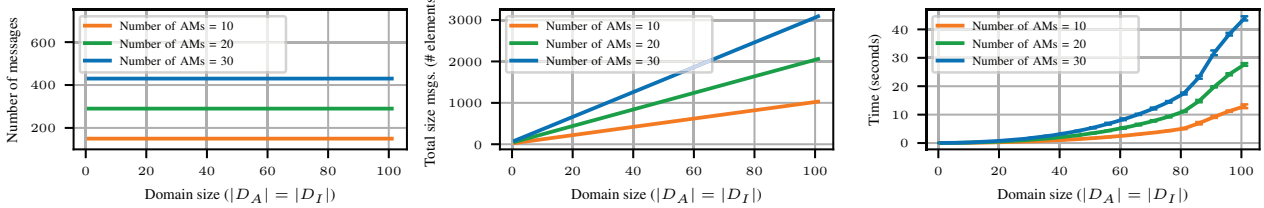

Fig. 8: Effect of increasing the domain sizes on coordination overheads. Error bars show three standard deviations.

the researchers' bias in extending Simdex, which we addressed by carrying out minimal changes to the exemplar and by using the original adaptation logics and dataset and 3) the choice of coordination objective, which we addressed by specifying the some goal components as random variables.

The main threat to the *external validity* is the choice of subject system, which we addressed by focusing on a load-balancing scenario (highly-relevant to the self-adaptive system community [3], [12], [32], [33]). Another threat is the choice of DPOP for (**RQ1**). However, because DPOP guarantees true optimal solutions, our results will hold for other algorithms offering similar guarantees, e.g., ADOPT.

VI. Related work

1) Coordination of adaptation planning strategies: In contrast to CoADAPT, existing techniques [23], [24], [26] for coordinating adaptation planning strategies are centralized. For example, [25], [26] propose a meta-manager employing a model checker to find the best assignment of adaptation planning strategies. Similar to us, [25], [26] frame coordination as an optimization problem, however only covering the centralized case and without differentiating between concerns.

2) Meta-adaptation: Assignment of adaptation planning strategies is essentially a special case of meta-adaptation [22], [29]. Many architectures and approaches for meta-adaptation have been proposed, however, in contrast to CoADAPT, they target a single self-adaptive system [22], [29] or employ centralized coordination [31].

3) Decentralized self-adaptive systems: A number of coordination techniques were proposed under the term of *decentralized self-adaptive systems* [2], [36]. However, these techniques are usually conceived from the perspective of a single administrative domain, thus they do not differentiate between local and shared concerns. Moreover, existing techniques also suffer from the other limitations mentioned in Section I, i.e. they employ a supervisor [6], [8], [9], [12], share private information [5], [10], [15], make unrealistic assumptions [7], or rely only on local decision-making [5], [7], [10].

4) Collective adaptive systems: There exist many coordination approaches for collective adaptive systems (e.g., see [37], [38]). These approaches focus on self-organization and dynamic coordination of systems that sporadically, spontaneously, and ephemerally collaborate at runtime. These systems coordinate as long as it serves their own interests, usually lacking a permanent common goal. In contrast, CoADAPT addresses the coordination among individual self-adaptive systems that cooperate and were assembled together at design-time to fulfill a common goal.

5) DCOP for SAS: There are only a few works mentioning DCOP for self-adaptive systems. In [39] the author suggests that DCOP could outperform existing designs for self-adaptive systems based on the monitor, analyze, plan, and execute (MAPE) loop. Unlike [39], we propose DCOP as a coordination mechanism rather than a replacement for MAPE loops. Additionally, while we offer detailed specification techniques and experimental evaluation, [39] covers the topic with limited depth. In their works [40], the authors briefly mention DCOP as a potential component of their architecture for multi-agent middlewares. Their work does not describe in detail how DCOP should be employed nor do the authors evaluate their solution. In terms of application-oriented papers, [41] uses DCOP to realize self-adaptation in the context of wireless charging and [42] uses DCOP for self-adaptation in decentralized manufacturing. In contrast to these works, CoADAPT provides an application-independent coordination technique.

VII. Conclusion and outlook

We introduced CoADAPT, a decentralized technique for coordinating multiple self-adaptive systems. CoADAPT specifies coordination as a constraint optimization problem and introduces two types of constraints: 1) preference constraints, expressing concerns local to a system, e.g., adaptation preferences, and 2) consistency constraints, expressing concerns shared by multiple systems, e.g., adaptation conflicts. The resulting specification, paired with a distributed optimization algorithm, ensures that local concerns remain private to a system during coordination, while shared concerns are only

known to other systems involved in those concerns. We have realized CoADAPT for the coordination of adaptation planning strategies, demonstrated its feasibility in a cloud computing exemplar, and analyzed experimentally its scalability.

As future work, we wish to focus on the design-time engineering of the coordination tasks. To this end, we plan to investigate privacy-sensitive protocols for constructing the constraint optimization problem in a decentralized and automated way. Moreover, we wish to evaluate CoADAPT in more settings, such as complex, multi-stakeholder systems that span the whole cloud – IoT computing continuum.

REFERENCES

[1] D. Weyns, *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*. John Wiley & Sons, 2020.

[2] D. Weyns et al., *On Patterns for Decentralized Control in Self-Adaptive Systems*, 2013, pp. 76–107.

[3] T. Chen and R. Bahsoon, "Self-Adaptive Trade-off Decision Making for Autoscaling Cloud-Based Services," *IEEE Trans. Serv. Comput.*, vol. 10, no. 4, pp. 618–632, 2017.

[4] R. Wohlrab and D. Garlan, "A negotiation support system for defining utility functions for multi-stakeholder self-adaptive systems," *Requirements Engineering*, vol. 28, no. 1, pp. 3–22, 2023.

[5] R. Calinescu, S. Gerasimou, and A. Banks, "Self-adaptive Software with Decentralised Control Loops," in *18th Int. Conf. on Fundam. Approaches to Softw. Eng. (FASE)*, 2015, pp. 235–251.

[6] C. Tsigkanos, I. Murturi, and S. Dustdar, "Dependable resource coordination on the edge at runtime," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1520–1536, 2019.

[7] A. Metzger, A. Bayer, D. Doyle, A. Sharifloo, K. Pohl, and F. Wessling, "Coordinated run-time adaptation of variability-intensive systems: an application in cloud computing," in *1st Int. Workshop on Variability and Complexity in Softw. Des. (VACE)*, 2016, pp. 5–11.

[8] L. Baresi, D. F. Mendonça, and G. Quattrocchi, "PAPS: A Framework for Decentralized Self-management at the Edge," in *17th Int. Conf. on Service-Oriented Computing (ICSOC)*, 2019, pp. 508–522.

[9] I. Gerostathopoulos and E. Pournaras, "TRAPPed in traffic? a self-adaptive framework for decentralized traffic optimization," in *14th Int. Symp. on Softw. Eng. for Adaptive and Self-Manag. Syst. (SEAMS)*, 2019, pp. 32–38.

[10] M. D'Angelo, M. Caporuscio, V. Grassi, and R. Mirandola, "Decentralized learning for self-adaptive QoS-aware service assembly," *Future Gener. Comput. Syst.*, vol. 108, pp. 210–227, 2020.

[11] D. Matusek, T. Kluge, I. Shmelkin, T. Springer, and A. Schill, "Adaptation consistency of distributed role-oriented applications based on the actor model of computation," *LNNS*, p. 35–60, 2022.

[12] F. Rossi, V. Cardellini, and F. L. Presti, "Hierarchical scaling of microservices in kubernetes," in *2020 IEEE Int. Conf. on Autonomic Comput. and Self-Organizing Syst. (ACSOS)*, 2020, pp. 28–37.

[13] E. Carlini, M. Coppola, P. Dazzi, M. Mordacchini, and A. Passarella, "Self-Optimising Decentralised Service Placement in Heterogeneous Cloud Federation," in *10th Int. Conf. on Self-Adaptive and Self-Organizing Syst. (SASO)*, 2016, pp. 110–119.

[14] Y. Guo, M. Pan, Y. Fang, and P. P. Khargonekar, "Decentralized coordination of energy utilization for residential households in the smart grid," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1341–1350, 2013.

[15] A. Ismail and V. Cardellini, "Decentralized Planning for Self-Adaptation in Multi-cloud Environment," in *Advances in Service-Oriented and Cloud Computing*, 2015, pp. 76–90.

[16] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "ADOPT: Asynchronous distributed constraint optimization with quality guarantees," *Artificial Intelligence*, vol. 161, no. 1-2, pp. 149–180, 2005.

[17] A. Petcu and B. Faltings, "A scalable method for multiagent constraint optimization," in *19th Int. Joint Conf. on Artif. Intell. (IJCAI)*, 2005, p. 266–271.

[18] F. Fioretto, E. Pontelli, and W. Yeoh, "Distributed constraint optimization problems and applications: A survey," *J. Artif. Intell. Res.*, vol. 61, pp. 623–698, 2018.

[19] F. Fioretto, W. Yeoh, and E. Pontelli, "A multiagent system approach to scheduling devices in smart homes," in *16th Conf. on Auton. Agents and MultiAgent Syst. (AAMAS)*, 2017, pp. 981–989.

[20] T. Léauté and B. Faltings, "Coordinating logistics operations with privacy guarantees," in *22nd Int. Joint Conf. Artif. Intell. (IJCAI)*, 2011.

[21] Z. Jin, J. Cao, and M. Li, "A Distributed Application Component Placement Approach for Cloud Computing Environment," in *9th Int. Conf. on Dependable, Autonomic and Secure Comput. (DASC)*, 2011, pp. 488–495.

[22] V. Lesch, M. Hadry, S. Kounev, and C. Krupitzer, "Self-aware optimization of adaptation planning strategies," *ACM Trans. Auton. Adapt. Syst.*, 2022, just Accepted.

[23] H. Wang and J. Ying, "An approach for harmonizing conflicting policies in multiple self-adaptive modules," in *2007 Int. Conf. on Machine Learning and Cybernetics*, vol. 4, 2007, pp. 2379–2384.

[24] N. Khakpour, R. Khosravi, M. Sirjani, and S. Jalili, "Formal analysis of policy-based self-adaptive systems," in *25th ACM Int. Symp. on Applied Computing*, 2010, p. 2536–2543.

[25] T. Glazier and D. Garlan, "An automated approach to management of a collection of autonomic systems," in *4th Int. Workshops on Found. and Appl. of Self* Syst. (FAS*W)*, 2019, pp. 110–115.

[26] T. J. Glazier, D. Garlan, and B. Schmerl, "Automated management of collections of autonomic systems," in *1st Int. Conf. on Autonomic Comput. and Self-Organizing Syst. (ACSOS)*, 2020, pp. 82–91.

[27] A. Molina Zarca, M. Bagaa, J. Bernal Bernabe, T. Taleb, and A. F. Skarmeta, "Semantic-aware security orchestration in sdn/nfv-enabled iot systems," *Sensors*, vol. 20, no. 13, p. 3622, 2020.

[28] M. Krulis, T. Bures, and P. Hnetynka, "Simdex: a simulator of a real self-adaptive job-dispatching system backend," in *17th Int. Symp. on Softw. Eng. for Adaptive and Self-Manag. Syst. (SEAMS)*, 2022, pp. 167–173.

[29] I. Gerostathopoulos, T. Bures, P. Hnetynka, A. Hujecek, F. Plasil, and D. Skoda, "Strengthening adaptation in cyber-physical systems via meta-adaptation strategies," *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 3, 2017.

[30] V. Braberman, N. D'Ippolito, J. Kramer, D. Sykes, and S. Uchitel, "An extended description of morph: A reference architecture for configuration and behaviour self-adaptation," *LNCS*, p. 377–408, 2017.

[31] B. R. Siqueira, F. C. Ferrari, and R. de Lemos, "Design and evaluation of controllers based on microservices," in *18th Int. Symp. on Softw. Eng. for Adaptive and Self-Manag. Syst. (SEAMS)*. IEEE, 2023, pp. 13–24.

[32] F. A. de Oliveira, T. Ledoux, and R. Sharrock, "A Framework for the Coordination of Multiple Autonomic Managers in Cloud Environments," in *7th Int. Conf. on Self-Adaptive and Self-Organizing Syst. (SASO)*, 2013, pp. 179–188.

[33] M. Nardelli, G. Russo Russo, V. Cardellini, and F. Lo Presti, "A Multi-level Elasticity Framework for Distributed Data Stream Processing," in *Euro-Par 2018: Parallel Process. Workshops*, 2018, pp. 53–64.

[34] P. Rust, G. Picard, and F. Ramparany, "pyDCOP: a DCOP library for Dynamic IoT Systems," in *18th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019, p. 5.

[35] M. Carwehl, T. Vogel, G. N. Rodrigues, and L. Grunske, "Runtime verification of self-adaptive systems with changing requirements," in *18th Int. Symp. on Softw. Eng. for Adaptive and Self-Manag. Syst. (SEAMS)*, 2023.

[36] F. Quin, D. Weyns, and O. Gheibi, "Decentralized Self-Adaptive Systems: A Mapping Study," in *16th Int. Symp. on Softw. Eng. for Adaptive and Self-Manag. Syst. (SEAMS)*, 2021.

[37] A. Diaconescu et al., "Architectures for collective self-aware computing systems," in *Self-Aware Comput. Syst.* Springer, 2017, pp. 191–235.

[38] A. Bucchiarone, "Collective adaptation through multi-agents ensembles: The case of smart urban mobility," *ACM Trans. Auton. Adapt. Syst.*, vol. 14, no. 2, oct 2019.

[39] C. Harold, "Towards self-managing systems through decentralised constraint optimisation," in *17th Int. Conf. on Practical Appl. of Agents and Multi-Agent Syst. (PAAMS 2019)*, 2019, pp. 334–338.

[40] A. V. Uzunov, M. Brennan, M. B. Chhetri, Q. B. Vo, R. Kowalczyk, and J. Wondoh, "AWaRE2-MM: A Meta-Model for Goal-Driven, Contract-Mediated, Team-Centric Autonomous Middleware Frameworks for Antifragility," in *28th Asia-Pacific Softw. Eng. Conf. (APSEC)*, 2021, pp. 547–552.

[41] C. J. v. Leeuwen, K. S. Yildirim, and P. Pawelczak, "Self Adaptive Safe Provisioning of Wireless Power Using DCOPs," in *11th Int. Conf. on Self-Adaptive and Self-Organizing Syst. (SASO)*, 2017, pp. 71–80.

[42] J. Hirsch, M. Neumayer, H. Ponsar, O. Kosak, and W. Reif, "Distributed Constraint Optimization for Task Allocation in Self-Adaptive Manufacturing Systems," in *2nd Int. Conf. on Autonomic Comput. and Self-Organizing Syst. Companion (ACSOS-C)*, 2021, pp. 62–67.