Learning to Learn in Collective Adaptive Systems: Mining Design Patterns for Data-driven Reasoning

Mirko D'Angelo*, Sona Ghahremani†, Simos Gerasimou§,
Johannes Grohmann**, Ingrid Nunes¶, Sven Tomforde‖, Evangelos Pournaras‡

*Linnaeus University, Växjö, Sweden, Email: mirko.dangelo@lnu.se
† Hasso Plattner Institute, Universität Potsdam, Potsdam, Germany, Email: sona.ghahremani@hpi.de
§University of York, United Kingdom, Email: simos.gerasimou@york.ac.uk
**University of Würzburg, Germany, Email: johannes.grohmann@uni-wuerzburg.de
¶Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil, Email: ingridnunes@inf.ufrgs.br
‖Christian-Albrechts-Universität zu Kiel, Germany, Email: st@informatik.uni-kiel.de
‡University of Leeds, United Kingdom, Email: e.pournaras@leeds.ac.uk

***Abstract*—Engineering collective adaptive systems (CAS) with learning capabilities is a challenging task due to their multi-dimensional and complex design space. Data-driven approaches for CAS design could introduce new insights enabling system engineers to manage the CAS complexity more cost-effectively at the design-phase. This paper introduces a systematic approach to reason about design choices and patterns of learning-based CAS. Using data from a systematic literature review, reasoning is performed with a novel application of data-driven methodologies such as clustering, multiple correspondence analysis and decision trees. The reasoning based on past experience as well as supporting novel and innovative design choices are demonstrated.**

***Index Terms*—collective adaptive systems, design pattern, multi-agent system, learning, data mining, reasoning, decision tree, clustering**

I. Introduction

Collective adaptive systems (CAS) are distributed systems comprising multiple heterogeneous agents. Each agent does not individually possess system-wide knowledge and can: (i) interact with other agents either directly or indirectly; (ii) exhibit learning capabilities to expand its personal knowledge; and (iii) make decisions based on collective or aggregated knowledge from its peers [1]–[3].

Employing agents with such characteristics allows constructing highly autonomous systems exhibiting self-adaptive properties. As a result, learning-based CAS can cope with uncertainties and adapt accordingly to fulfill their requirements and improve their performance and reliability.[1]

Engineering learning-based CAS is complex due to their non-deterministic and highly dynamic operational environment, emerging from simultaneous interactions of several autonomous entities. Moreover, system-wide knowledge is distributed among the agents, entailing that advanced mechanisms should be used for efficient knowledge acquisition and sharing. Finally, the use of learning adds yet another layer of complexity, influenced by the availability of data, choice of technique, model instantiation, and model update. Thus, the numerous influencing decisions emerging from such a multi-dimensional and the complex design space perplex the engineers' choices when designing learning-based CAS.

With the growing availability of software engineering data, data-driven techniques have emerged as an effective methodology to provide software practitioners with up-to-date and pertinent information supporting the decision-making process [4]. Data-driven methodologies support dimensionality reduction by recognizing correlations in data [5] and have been extensively applied in the literature, e.g., to understand software evolution [6] or to discover instances of design patterns from the system's source code [7].

Collecting data about relevant past experiences, extracting knowledge from it, and making the knowledge available in a manner that can be reasoned upon is a first step towards supporting CAS engineers in navigating through the large design space of such systems and making cost-effective choices [8].

This paper contributes to this direction in the following ways. (i) We extend our previous systematic literature review on learning-enabled CAS [9] and use the analysis results as data capturing relevant past experiences. (ii) By employing data-driven methodologies, we identify correlations in the collected data, based on which we present relevant design-time reasoning knowledge in the form of design guidelines. And (iii) we structure the data (i.e., past experiences) as a decision tree representing a reasoning knowledge that can serve either as a design-time recommender or to spot design gaps.

The paper is organized as follows. Section II introduces the data acquisition process and the data-driven methodologies. Section III presents the design guidelines elicited from the analysis and shows the knowledge as a decision tree. Section IV concludes the findings and discusses future work.

II. Methodology

In this section, we present how we extended our systematic literature review on learning-enabled CAS [9] and use its results as input for our data-driven methodologies. Then we

[1] http://www.focas.eu/manifesto/

K	Vector of clusters stabilities	Times each cluster is dissolved in 100 re-sampling
2	h0.90,0.85i	h0,0i
3	h0.82,0.86,0.50i	h2,0,59i
4	h0.87,0.86,0.60,0.61i	h0,1,48,9i
5	h0.84,0.77,0.64,0.69, 0.27i	h0,8,43,29,92i
9	h0.70,0.69,0.61,0.64,0.84,0.74,0.47,0.51,0.33i	h18,32,40,36,21,29,67,69,93i

(a) Silhouette values

(b) Bootstrapping results

(c) K = 2

(d) K = 3

(e) K = 4

(f) K = 9

(g) Autonomy

(h) Emergent behaviour

(i) Cooperative agent

(j) Trigger rst

(k) Trigger update

(m) Knowledge access

(l) Behaviour

(o) Domain

(n) Technique

Fig. 1: HAC results: evaluation (a-b), choices of K (c-f), design dimensions mapped to HAC (g-o).

Table I: DIMENSIONS OF LEARNING-BASED CAS

Dimension	Description
Application Domain	The domain for which a CAS is developed
Autonomy	The agents' ability to act autonomously or in need of a supervised entity
Knowledge Access	The amount of information available to an agent from its peers or the environment
Behaviour	Agent's comportment toward self-goals and system-wide goals
Emergent Behaviour	Whether the collective demonstrates a behaviour different than the one of single agents
Cooperative Agent	Whether agents are cooperating
Learning Technique	The technique used by agents to exhibit learning
Trigger First	The initial knowledge used to instantiate learning models
Trigger Update	Criteria for updating the learning models

explain the two employed data-driven methodologies, i.e., Hierarchical Agglomerative Clustering (HAC) and Multiple Correspondence Analysis (MCA).

Clustering allows us to capture the set of design choices applied on CAS based on past experience (i.e., the state of the art). The rationale is that clustering analysis can be used to detect design patterns and reduce the complexity of the design space. Correspondence analysis, on the other hand, focuses on identifying the correlation between the design dimensions. This allows capturing system constraints that impose certain design choices. As a result, the engineer obtains better insights about the interplay and interactions between different design dimensions. The relevance and effectiveness of these techniques for system design decisions has been shown in [10].

A. Data Acquisition

In [9], we conducted a systematic literature review of 52 studies related to learning-based CAS. The investigated papers are classified based on their choices for the nine design dimensions envisioned for learning-enabled CAS. Table I enumerates these nine design dimensions, which are used to define the design space of CAS. These dimensions are the result of a thorough discussion and analysis of the domain at the GI-Dagstuhl seminar on Software Engineering for Intelligent and Autonomous Systems [11].

This paper extends the literature review of [9] by following the same search and analysis methods. The analysis of the most recent studies added 7 additional research papers. As a result, 59 studies are included in our updated systematic literature review[2]. A vector including the nine design choices of a paper (see Table I) constitutes a data point for our data-driven analysis. Accordingly, the considered dataset has 59 data points.

B. Hierarchical Agglomerative Clustering (HAC)

As a starting point, HAC [12] considers each observation (i.e., each reviewed paper in our case) as a separate cluster. Then, HAC incrementally identifies and merges the two most similar clusters. The notion of similarity between papers refers to similarity between their design dimensions (see Table I). Since the dimensions identified in our systematic literature review are categorical, we adopt the Gower Distance [13] measure, which is a simple but widely applied distance metric suitable for categorical data. In particular, the distance $d(i;j)$ for any pair of inputs x_i and x_j across the examined number of dimensions M is given by $d(i;j) = 1=M \sum_{m=1}^{M} d_{ij}^m$ where $d_{ij}^m = 0$ if $x_i^m = x_j^m$; and $d_{ij}^m = 1$ otherwise. Evaluating other distance metrics is outside the scope of this paper.

A typically applied method for identifying the desired number of final clusters in HAC (given by K) involves using clustering validation criteria [14]. In our analysis, we rely on the silhouette values [15] and the bootstrap method [16]. The results of clustering using this method are depicted in Figure 1.

The silhouette value is a measure of data consistency, where higher values represent higher coherence between the data points within a cluster. Figure 1a shows the silhouette values for HAC up to 10 clusters. The plot suggests that after five clusters, the consistency of data points within each cluster drops and reaches a local maximum with $K = 9$.

The bootstrap analysis enables the assessment of the stability of the considered number of clusters by investigating how easily clusters dissolve. To conduct the bootstrap analysis, we select the clusters with average silhouette width greater than or equal to 0.18, i.e., the average silhouette width with $K = 9$ (see the dotted line in Figure 1a). Lower silhouette values result in weak or artificial structures. The results of the bootstrapping are presented in Figure 1b. For each number of considered clusters K we report: the vector of cluster stabilities (values close to 1 indicate stable clusters) and the number of times each cluster is dissolved after 100 resampling (clusters that are dissolved often are unstable). The results suggest that $K = 2$ forms two stable clusters that are never dissolved. $K = 3$ and $K = 4$ introduce mild degrees of instability, while $K = 5$ and $K = 9$ result in high instability.

Figures 1c–1f show how the reviewed papers, distinguished by their id, are partitioned in 2, 3, 4, and 9 clusters, respectively. In particular, each figure shows how cluster partitions, identified with different colors, map to the dendrogram tree generated by the HAC. For instance, Figure 1c depicts the result of clustering for $K = 2$, where the two clusters are identified with blue (31 papers) and red (28 papers).

Figures 1g–1o illustrate the results of clustering the dataset based on each design dimension introduced in Table I. In particular, each figure shows how the concrete values of a specific design dimension partition the papers in the dataset. Different colors in each dendrogram represent a value for the considered design dimension. In Figure 1i, for instance, the dendrogram partitions the papers according to the design dimension Cooperative Agent. The paper ids (i.e., data points)

[2] The updated review and replication package of the analysis can be seen at: https://mi-da.github.io/learning-to-learn-CAS/

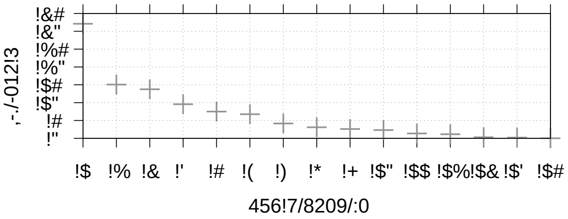


Fig. 2: Variance retained by MCA dimensions.

in red employ cooperative agents while the paper ids in black exclude cooperation among the employed agents.

The partitions identified by the design dimension Cooperative Agent in Figure 1i are similar to the those depicted by the clustering analysis with $K = 2$ in Figure 1c (except for paper 40). Similarly, mapping the results of the clustering based on the design dimensions and the results of HAC can be leveraged to capture the relevant set of design choices for CAS based on past experience.

C. Multiple Correspondence Analysis (MCA)

MCA is a generalization of the principal component analysis (PCA) for categorical data, which aims to summarize the underlying structures in the fewest possible dimensions [17]. In particular, MCA identifies new latent dimensions, which are a combination of the original dimensions and hence can explain information that is not directly observable.

Figure 2 depicts the variance of the new dimensions (i.e., principal components) identified by MCA after applying the optimistic Benzécri correction [17]. The larger variance of the dimensions indicates capturing more meaningful correlations by the considered dimensions.

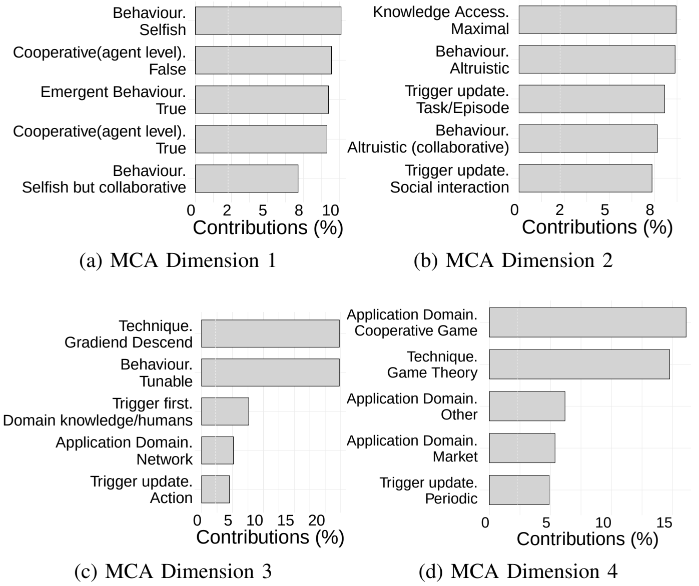
In order to have a good intuition of the MCA results, it is necessary to choose the number of components to retain and observe how the design dimensions of CAS map to the new identified dimensions. Following the average rule introduced by Lorenzo-Seva et. al. [18], we kept all the dimensions with variance greater than 7% (i.e., four dimensions are retained). Figure 3 depicts the contributions (in percentage) of the design dimensions' values to the definition of the MCA dimensions. Figures 3a–3d show only the five most contributing variables, as small contributions imply low relevance. The dashed lines represent the expected contribution if all the values of the design dimensions would contribute equally to the definition of the MCA dimension.

III. Application

We first present the observations and guidelines emerging from applying our data-driven analysis. Then, we present the extracted reasoning knowledge in the form of a decision tree that can be used both as a recommender system and as a mechanism to identify design gaps.

A. Observations and Guidelines

Emergent behaviour and cooperative agents are two of the most important design dimensions identified by MCA (see Figure 3a) that promote the formation of two stable clusters (see Figure 1c).



(a) MCA Dimension 1

(b) MCA Dimension 2

(c) MCA Dimension 3

(d) MCA Dimension 4

Fig. 3: Contribution of design dimensions to MCA dimensions.

Our analysis highlights that designing non-cooperative (see Figure 1i) and selfish agents (see Figure 1l) often results in a collective emergent behaviour (see Figure 1h).

In contrast, when interactions are introduced, it is often the case that designers have a target system behavior in mind. This is derived from the observation that systems using altruistic approaches or adopting a certain level of collaboration are characterized by employing cooperative agents and no emergent behaviour.

MCA finds correlation between the two dimensions indicating the triggers for instantiating (i.e., trigger first) and refining the learning models (i.e., trigger update). The process employed for learning a new or updating the existing model is a major factor affecting the ability of agents within CAS to operate in uncertain environments.

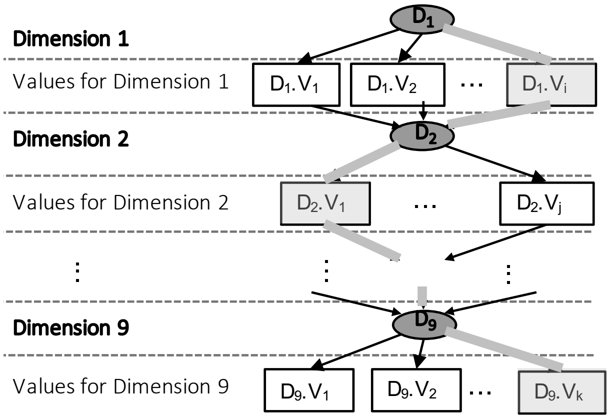


Fig. 4: Decision tree representation of reasoning knowledge.

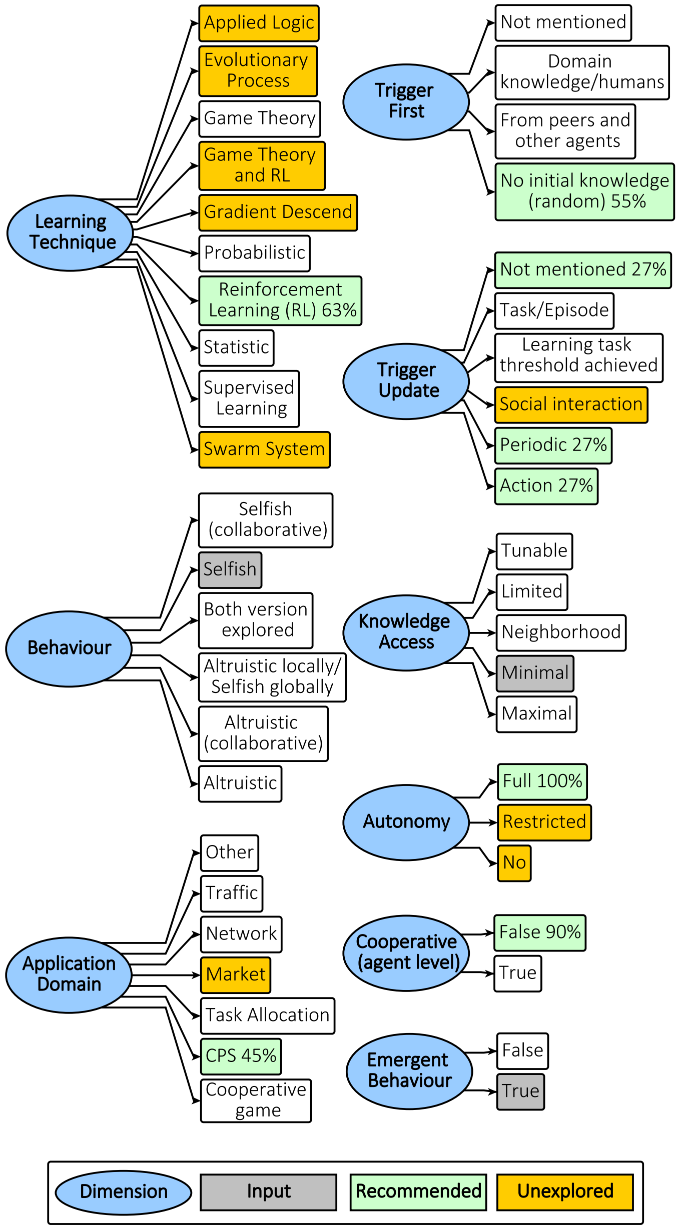


Fig. 5: Design dimensions as decision nodes.

Our analysis suggests that in the presence of maximum knowledge access (see Figure 1m), domain or human knowledge is often used to instantiate the learning model and the refinement is typically through episodes or task-based (see Figures 1j–1k and Figure 3b). On the other hand, we identify a group of systems that are characterized by minimal or neighborhood knowledge access and, as a result, their learning models operate under high uncertainty. In this case, the learning model is typically instantiated without any prior knowledge and the refinement is threshold-based or via social interactions. Our analysis highlights how agents often rely on their peers' knowledge in the presence of learning uncertainty to construct a bigger picture of the environment as a means to tackle uncertainty.

We summarize our findings by proposing the following design guidelines, which are the result of how existing CAS are usually engineered.

Design for System-wide vs. Agent-level Goals. If the objective of the CAS is to fulfill a system-wide goal, then: *(i)* cooperative agents shall be used; *(ii)* agents should exhibit altruistic or collaborative behaviour; and *(iii)* there should be a degree of knowledge exchange among agents (i.e., minimal level of knowledge access should be avoided). The engineer should decide/reason about the trade-off between the desired level of knowledge and the introduced cost in terms of performance.

In contrast, designing a CAS with agent-level objectives in mind: *(i)* eliminates the need for employing cooperative agents; *(ii)* agents demonstrating selfish behaviour (hence, prioritizing agent-level goals) should be employed; and *(iii)* knowledge access among agents is not required since no coordination is needed. In such a scenario, the engineer can expect the (implicit) system-wide goal to be fulfilled as an emergent behaviour.

Access to Training Data/ Domain Knowledge. The choice of a learning technique can be greatly affected by the availability of training data or domain expert knowledge for model instantiation and update. When access to sufficient data for model-based learning techniques is limited, model-free techniques should be employed. When no sufficient data is available, the refinement should be threshold-based or via social interactions. Domain expert knowledge can be leveraged to set the model refinements trigger to episodic or task/action-based. Finally, the choice of the learning technique should be independent of the application domain of interest as we observe no correlation between these two design dimensions.

B. Mining the Knowledge

We employ decision tree modeling [19] to generate a decision tree from the collected data. The identified design dimensions introduced in Table I form the nine decision nodes of the tree as depicted in Figure 4. Each dimension D_i is further decomposed into multiple values $D_i.V_j$. We collected the values for each design dimension from our survey in [9]. Figure 5 expands the design dimensions introduced in Table I to their different values.

Representing the reasoning knowledge as a decision tree provides a top-down scheme to explore the design space of learning-based CAS for the CAS designers. The decision tree can be traversed starting from the root node (i.e., D_1 in Figure 4) and making decisions among the available choices for each dimension D_i, (i.e., $D_i.V_j$ in Figure 4) until all the dimensions are visited. The highlighted trajectory in Figure 4 shows an exemplary path in the decision tree where value V_i is selected for dimension D_1, value V_1 is chosen for D_2, and for dimension D_9 value V_k is selected. The path can be summarized as $(D_1.V_i, D_2.V_1, ..., D_9.V_k)$.

Each dimension D_1–D_9 in Figure 4 is mapped to a design dimension introduced in Table I and is depicted in Figure 5