The Journal of Systems and Software 134 (2017) 12–28

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Self-organizing multi-agent systems for the control of complex systems

Jérémy Boes[*], Frédéric Migeon

IRIT, University of Toulouse 118, route de Narbonne, F-31062 Toulouse Cedex 9, France

ARTICLE INFO

Article history:
Received 30 September 2016
Revised 28 June 2017
Accepted 18 August 2017
Available online 30 August 2017

Keywords:
Multi-Agent systems
Control
Self-organization
Complex systems
Internal combustion engines

ABSTRACT

Because of the law of requisite variety, designing a controller for complex systems implies designing a complex system. In software engineering, usual top-down approaches become inadequate to design such systems. The Adaptive Multi-Agent Systems (AMAS) approach relies on the cooperative self-organization of autonomous micro-level agents to tackle macro-level complexity. This bottom-up approach provides adaptive, scalable, and robust systems. This paper presents a complex system controller that has been designed following this approach, and shows results obtained with the automatic tuning of a real internal combustion engine.

1. Introduction

Controlling a system means being able to perform the adequate modifications on its inputs in order to set the outputs on a desired state. Over the course of History, humans made tremendous efforts to control systems that are more and more complex: nonlinear, dynamic, noisy, with a large number of inputs and outputs, and so on. Yet, the law of requisite variety (Ashby, 1956) implies that the complexity of a controller has to be greater than or equal to the complexity of the target system. Thus, the design of a controller involves the design of a complex system. This is a challenge for engineering.

Complexity is often tackled a posteriori, to study existing systems. On the contrary, methods enabling the design of complex systems that meet strict requirements are quite rare. The main feature of a complex system is that its behavior can not be easily predicted (Heylighen, 2008). Usual design methods, for instance in software engineering, seek to a priori eliminate any unexpected event. The design process must ensure that everything will be smooth at runtime. But, as any other complex system, complex programs sometimes have unexpected, unpredictable behaviors, and these classical methods fail.

For instance, in the field of system control, the usual methods in the industry rely on the construction of a fine mathematical model of the target system, that is later used to compute the commands to perform, given some set points. The cost and difficulty of the construction (and the tuning) of a mathematical model is high. An often used alternative is machine learning. Giving the ability to learn to a controller enables it to learn the behavior of the target system and build a model from data. However, this method shows its limits when used with complex systems. Nonlinearities in the learnt model lead to overcostly or impossible computations in the control system. Another possibility exists: directly learning the adequate commands, instead of a model that will later lead to the said commands. We then focus only on the inputs and outputs of the controlled system, without trying to decipher its internal mechanisms.

Another difficulty is scalability. While various control methods exist, they (almost) all fail to scale when a large number of inputs and outputs are involved. Most advanced solutions rely on the distribution of the control. Instead of letting a central controller handle all the inputs, each input is controlled by one local controller, and all controllers try to cooperate to control the whole system.

Multi-Agent Systems (MASs), composed of autonomous entities, are naturally distributed. They can be very useful to the problem of the control of complex systems, for instance with multi-objective optimization (Khamis and Gomaa, 2014). Moreover, they bring innovative design methods. In particular, Adaptive Multi-Agent Systems (AMASs) are designed to be able to self-adapt at runtime to any unexpected event. Instead of wasting time trying to cope with any possible event during the design phase, we let the system deal with the unexpected at runtime. Driven by cooperation principles, agents self-organize locally to produce and maintain the desired global function.

[*] Corresponding author.
E-mail addresses: boes@irit.fr (J. Boes), migeon@irit.fr (F. Migeon).

http://dx.doi.org/10.1016/j.jss.2017.08.038
0164-1212/© 2017 Elsevier Inc. All rights reserved.

This paper presents experimental results obtained with an AMAS designed to control complex systems, and applied to the calibration of real heat engines. This system is fully described in English for the first time in this paper. Able to learn and control simultaneously, it provides a generic and robust solution to the problem of control. It is a good example of the ability of AMASs to be efficient in real life conditions.

Section 2 gives a quick background on control. Section 3 introduces our approach and Section 4 presents our system. Results, obtained in simulated as well as in real conditions, are showed in Section 5. Section 6 concludes with our perspectives.

2. Related works

Our work is at the crossroad of the fields of complex systems, control, and machine learning. It is inspired by the ideas of Edgar Morin on complexity (Morin, 2008), which we apply here to the design of self-adaptive control systems.

2.1. Complex systems

The notion of complexity reflects the difficulty to analyze a system and to forecast its behavior. Nonlinearities, inner feedback loops, large number of inputs/outputs/inner parts, uncertainty on the measures, and unpredictable behaviors are some of the recurring features of complex systems. However, there is no common agreement on a definition. For instance, Kolmogorov defines the complexity of a string as the length of the shortest description of said string (Kolmogorov, 1998). While it is largely accepted, this measure implies that a purely random string is of maximal complexity, as it can only be described by its full enumeration. However this contradicts one of the key features of complexity: it is situated somewhere between total order and total chaos (Heylighen, 2008). Moreover, a complex system is dynamic, it is able to spontaneously change its state. It is important not to neglect this aspect during the analysis or the design of a system. Measures such as Kolmogorov complexity give too much attention to static, structural features of systems, and not enough to their dynamics. To this end, *dynamical depth* is based on the idea that the degree of complexity of a system is not given by its part and their causal relations, but by the imbrication of the different dynamics that drive its behavior (Deacon and Koutroufinis, 2014).

Furthermore, the general system theory states that the classical analytical approach can only be applied on systems whose parts are linear and share negligible interactions (Von Bertalanffy, 1968). This lets a lot of systems out of its scope, in particular complex systems. We need to follow a different approach than the reductionist top-down analysis for complex systems control as well as for complex systems design. The Adaptive Multi-Agent Systems theory is being developed in this regard.

2.2. Control

Control approaches also find their limits when faced with complexity. Artificial Intelligence (AI), and in particular machine learning, are used to overcome these limits.

The objective with AI in control is to automatically learn either the model of the target system, the tuning of the model, the calibration of the controller, or directly control laws from observations. For instance, Jesus and Barbosa (2013) uses a genetic algorithm to learn the optimal tuning of PIDs. This approach gives excellent results but needs a large number of iterations. Moreover, if the behavior of the controlled system changes over time (for instance, because of mechanical wear), the tuning must be entirely redone, it is not adaptive.

The biggest difficulty of dual control is to find the correct balance between probe actions and control actions. A way to do this is to use neural networks to learn this balance from data (Fabri and Bugeja, 2013). This approach is limited to control affine systems, i.e. systems that reacts linearly to modifications on their inputs.

The most promising approach for scaling up, i.e. for controlling a large number of inputs with many criteria on many outputs, is to distribute the control. For instance, Bull et al. (2004) and Choy et al. (2006) control road traffic junction signals on several crossroads. In these approaches, there is no central controller that handles all the traffic junctions, each crossroad is controlled by a local controller. Bull et al. (2004) uses learning classifier systems, while Choy et al. (2006) uses a combination of neural networks, genetic algorithms and fuzzy logic. They obtained very interesting results, but the difficulty to instantiate their approaches to real life problems is a severe drawback.

Our approach uses feedback loops to learn not the model of the controlled system but the control laws themselves, and distributes a controller on each controlled input. Inner feedback loops ensure an adequate balance between exploration and exploitation of the model.

2.3. Machine learning

A program learns when it is able to improve its functionality using its experience, i.e. data acquired during its execution (Mitchell, 2006). Machine learning has been heavily influence by the way we think the human mind works. The two well-known methods for machine learning are supervised learning and unsupervised learning, whether examples of the expected results are presented to the learning program or not. However, this distinction is merely technical and does not allow to highlight the fundamental differences between machine learning algorithms. We prefer the following five categories: Behaviorism, Cognitivism, Connectivism, Evolutionism, and Constructivism.

Behaviorism considers the learner as a black-box. Learning occurs when the observed behavior changes in response to the dynamics of the environment. In machine learning, the behavior is then a product of the initial state of the program and its progressive conditioning by its environment through a feedback loop. Reinforcement learning can be considered as a behviourist machine learning approach. It is notably popular in robotics (Kober et al., 2013). Its most well-known algorithm is Q-Learning (Watkins and Dayan, 1992).

On the contrary, cognitivists consider that what is important is not what the learner does but what he knows. Cognitivist machine learning algorithms classically rely on symbol manipulation, and thus on a predefined set of symbols, which is not adequate when dealing with complexity (Raghavan et al., 2016).

Connectionism considers learning at a lower level in the brain: the dynamic interconnection of neurons. In machine learning, it regroups all the artificial neural network algorithms, from back-propagation perceptrons to the more recent Kohonen maps (Astudillo and Oommen, 2014) and deep learning algorithms (Deng and Yu, 2014). They show impressive results but need a huge amount of data and computing power.

Evolutionism considers learning at the scale of a species rather than an individual. Evolutionary algorithms evolve a population of solutions towards better solutions by evaluating them, mutating them, and crossing the best individuals. These algorithms are interesting because they can tackle problems for which there is no known solution, but they are time-consuming and the fitness function can be difficult to obtain (Bongard, 2013).

Constructivism is the idea that humans have the ability to construct knowledge in their own mind through interactions with the environment. Constructivist artificial intelligence aims at designing