Evolution as a Tool to Design Self-organizing Systems

István Fehérvári and Wilfried Elmenreich

Institute for Networked and Embedded Systems,
University of Klagenfurt
{*forename.surname*}@aau.at

Abstract Self-organizing Systems exhibit numerous advantages such as robustness, adaptivity and scalability, and thus provide a solution for the increasing complexity we face within technical systems. While they are attractive solutions, due to their nature, designing self-organizing systems is not a straightforward task. Artificial evolution has been proposed as a possible way to build self-organizing systems, but there are still many open questions on how an engineer should apply this method for this purpose. In this paper we propose a system architecture for evolving self-organizing systems, that marks the major cornerstones and decisions the designer has to face, thus providing a practical set of guidelines.

Keywords: self-organizing systems, evolutionary design, emergence, complexity

1 Introduction

Self-organizing systems are inherently scalable, adaptive and tend to be very robust against single-point failures, which made them an appealing choice in the technical domain [1]. Typically, a design procedure of such systems would be to find the local, micro-level interactions that will result in the desired global behavior. Unfortunately, there is no straightforward way for the design of these rules yet so that the overall system will show the desired macro-level properties.

There exist different approaches to design self-organizing systems. Gershenson [2] introduces a notion of "friction" between two components as a utility to design the overall system iteratively. Auer et al. [3] deduce the local behavior by analyzing a hypothetical omniscient "perfect" agent. The usefulness of evolutionary algorithms to evolve cooperative behavior is demonstrated by Quinn et al. [4] by evolving teamwork strategies among a set of robots. The examples are not limited to robotic applications; Arteconi [5] applies evolutionary methods for designing self-organizing cooperation in peer-to-peer networks. A generic, universal approach to tackle the aforementioned design issues is to use some kind of an automated search, namely evolution to find the right set of local rules that will *drive* the system into the desired macro behavior.

In the domain of mathematical optimization or artificial intelligence metaheuristics, evolutionary methods have been already proven to be especially useful and so are still widely applied. Emergence and evolution have been studied by many scientists from different disciplines, like biology [6], mathematics [7], physics [8], social systems [9], and economic systems [10]. However, most work describes mechanisms, but does not give answers as to how to use those mechanisms to achieve an intended technical effect.

With respect to this question, the work by Belle and Ackley on evolvable code [11] and the work on learning for multi-agent systems [12] by Shoham and Leyton-Brown are of special interest for the research goals of the proposed project.

The idea of using this technique to design self-organizing systems has already been mentioned [13], however a deeper, step-by-step guide tailored for this domain is still missing. In this paper, we are presenting a practical set of guidelines in order the fill this gap and support engineers and scientist using evolutionary design.

In the following section, we give reference to related work on design of self-organizing systems with a special focus on evolutionary design. The basic idea of using evolution as a design tool is introduced in Section 2. In Section 3, we present a system architecture for evolving self-organizing systems that marks the major aspects of the approach. In the following subsection, we discuss each aspect of our design approach and introduce guidelines for practical application. The robot soccer case study described in Section 4 shows the application of the mentioned principles in practice. The paper is concluded in Section 5.

2 Design by Evolution

Typically, problems based on bio-inspired principles such as self-organization call for bio-inspired solutions. An approach would be to cope with the high complexity is to use some kind of guided automated search, namely evolutionary computation.

The main idea is to create an appropriate system model with its environment, where the components are represented in such a way, that their individual behaviors (i.e. set of local rules) can be iteratively improved via a heuristic algorithm. In other words, the components have to be *evolvable*.

Although the concept of evolutionary design is fairly simple, when it comes to application, it starts to become increasingly complex. In general, the whole process can be decomposed into three major steps: modeling the system and its environment; an iterative search, that explores new solutions; and a final validation phase (see Figure 1).

From an engineering point of view, the modeling phase is the upmost priority, where the crucial decisions have to be made in order to obtain useful results. Among many things this includes selecting the right computation model, the way of interactions and the whole evolutionary model (genotype to phenotype mapping, search algorithm, etc.). The next step is to let the whole system run on its own, thus gradually developing a viable solution with no or very little user interaction. The last phase for any design process is to verify the obtained results. Usually, this means a set of white or black box testing, but in the case of complex networked systems, novel approaches are necessary. Fortunately, there is a lot of ongoing research on how to evaluate such systems [14].

3 System Architecture

As mentioned in the previous section, building a proper system model is essential for a successful evolutionary design. In this paper we propose a system architecture for the evolutionary design of self-organizing systems, that identifies the major decisions and considerations that the designer must face during the modeling process. As can be seen in Figure 2, we distinguish 6 major components:

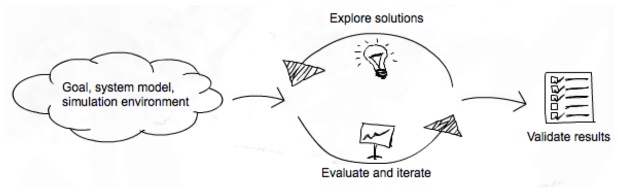

Figure 1: Basic flowchart of the evolutionary design

- The task description
- The simulation setup
- The interaction interface
- The evolvable decision unit
- The search algorithm
- The objective function

Typically, every engineering problem starts with a task description that properly identifies the objectives and constraints in detail. In other words, this can be seen as a contract that describes the expectations of the desired system on a high abstraction level. Consequently, the task description has a high influence on the applied simulation model and naturally on the objective function.

The next step is then to specify a system model based on the task description that is not only efficient, but accurately represents the important aspects of the system to be modeled while abstracting unnecessary factors. However, this step should not include how the components are represented or the way they interact with each other, since these would involve several further important decisions. For this purpose we separate them into an interaction interface unit and to an evolvable decision unit. In the former, one should plan the way the components of the system can interact with each other and their environment, thus not only defining the communication possibilities (i.e. sensors), but also their underlying interfaces (i.e. protocols).

The latter is focused on the actual representation of the components of the system. This might include one or more types of models depending on the homogeneity of the

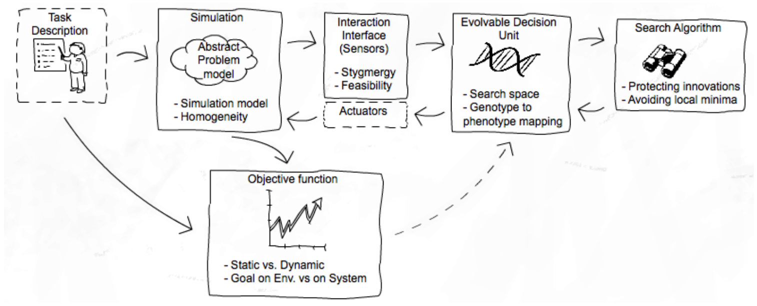

Figure 2: Proposed system architecture

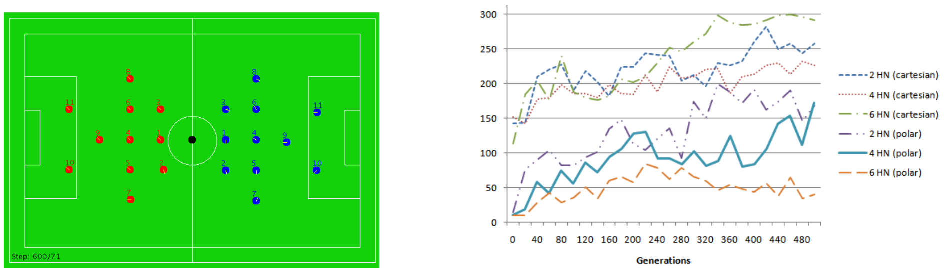

Figure 3: Robot soccer game simulation starting situation erformance for different interface and decision unit configurations over the number of generations. Interface design has significant impact, while size of neural network does not. Plots refer to evolved fully meshed neural networks. Performance is measured by matching teams against each other in a full tournament.

system. The reason why this part is separated from the system model is that the evolutionary method requires *evolvable* representations. There is, however, a vast set of such representations out there with their own advantages and disadvantages, so a careful decision has be made here.

In order for the evolutionary process to be operational there must be a valid and efficient search algorithm that iteratively optimizes the candidate solutions. While theoretically it can be completely separated from the representation of the evolvable components, it has to be noted that choosing the right algorithm has a significant overall effect on the convergence speed and on the quality of the results.

Last but not least the driving force of evolution must be also designed. While a good objective function has a key role in the process, there are unfortunately no guidelines available as to make a good function that not only ensures the desired behavior, but also makes the system avoiding unwanted states.

4 Case Study: Simulated Robot Soccer

The following case study describes an application of our design approach for a multi-robot scenario. The main focus lies on the aspects of the respective design methodology, the particular results of the case study are described separately in [15].

In this case study, an algorithm for simulated robot soccer teams was sought. Robot soccer players should learn to successfully play soccer without being given explicit instructions as to how a soccer team should behave. The robots all have the same hardware, and within a team, each robot has the same neural network. With respect to our design methodology, the following aspects have been tackled in the modeling process:

Simulation Model: The simulation model was based on existing abstractions from real robot soccer to provide a playground for a simulated robot soccer league. This model abstracts over mechanical aspects like slacking wheels or sophisticated sensor models for ball detection, while still providing a realistic interface for testing a distributed algorithm for making the robots play soccer. In particular, our simulation was based on the Robocup Soccer Simulation League without the features for centralized control via a coach.

Interaction Interface: Experiments with different set-ups showed that the sensor interface needs to be reduced to a few vectors indicating the position of the ball, the closest opponent and the closest teammate relative to the robot. In order to score a goal, the robots need to know where they are relative to the goal, so we added sensors indicating their relative field position. We tested the performance of evolved teams using a cartesian and a polar coordinate system representing the vectors. Results show that the cartesian coordinate system can be evolved to a better solution.

Decision Unit: Since there exists no particular theory, on how an evolvable decision unit for a soccer robot should be constructed, we experimented with several types of neural networks and varied the network size. Results indicate that a recursive neural network model, though being harder to analyze for a human observer, provides better capabilities than a feed forward model. The network size had less impact on the results than expected – solutions with 2, 4 or 6 neurons did not differ significantly in their performance (see Figure 3.

Objective Function: Playing soccer involves complex coordination of all team members based on variables of high dynamics like the position and momentum of the ball and all players. Therefore, we identified separate performance indicators, namely field coverage, ball possession, playing the ball actively, getting the ball near the opponents goal and scoring goals. Putting these factors hierarchically into an objective function enabled a smooth evolution of game-play [1]

A thorough description of the results is given in [15]. The robot soccer experiment was conducted with the tool FRamework for Evolutionary Design (FREVO) [16]. [2].

5 Conclusions and Future Work

This paper discusses the basic building blocks for applying evolutionary algorithms to the design of self-organizing systems for technical applications. Our survey of existing literature has revealed that despite the many reports on applying this approach to generate a self-organizing system, there is a need for a generic description of the underlying system engineering process. We therefore introduced several vital cornerstones of such an approach. In particular, we identified the task description, simulation setup, interaction interface, decision unit, search algorithms and fitness function.

The paper makes two main contributions: It supports the design process by defining the system aspects that must be considered. Its second contribution is that this approach may provide the basis for a qualified engineering process for self-organizing systems in technical applications. In future work we will elaborate our proposed methodology with detailed guidelines and principles for task description, simulation setup, interaction interface, decision unit, search algorithms and fitness function.

[1] see `http://www.youtube.com/watch?v=cP035M_w82s` for a video demo.

[2] FREVO including the robot soccer simulation described above are available as open source at `http://frevo.sourceforge.net/`

Acknowledgments

This work was supported by the Austrian Science Fund (FFG) grant FFG 2305537 and the Lakeside Labs GmbH, Klagenfurt, Austria, and funding from the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant KWF 20214|21532|32604, and KWF 20214|22935|34445. We would like to thank Lizzie Dawes for proofreading the paper.

References

1. W. Elmenreich and H. de Meer. Self-organizing networked systems for technical applications: A discussion on open issues. In J.P.G. Sterbenz. K.A. Hummel, editor, *Proceedings of the Third International Workshop on Self-Organizing Systems*, pages 1–9. Springer Verlag, 2008.
2. C. Gershenson. *Design and Control of Self-organizing Systems*. PhD thesis, Vrije Universiteit Brussel, 2007.
3. C. Auer, P. Wüchner, and H. de Meer. A method to derive local interaction strategies for improving cooperation in self-organizing systems. In *Proceedings of the Third International Workshop on Self-Organizing Systems*, Vienna, Austria, December 2008.
4. M. Quinn, L. Smith, G. Mayley, and P. Husb. Evolving teamwork and role allocation with real robots. In *Proceedings of the 8th International Conference on Artificial Life*, pages 302–311. MIT Press, 2002.
5. S. Arteconi. Evolutionary methods for self-organizing cooperation in peer-to-peer networks. Technical Report UBLCS-2008-5, Department of Computer Science, University of Bologna, 2008.
6. M. C. Donaldson, M. Lachmann, and C. T. Bergstrom. The evolution of functionally referential meaning in a structured world. *Journal of Theoretical Biology*, 246:225–233, 2007.
7. F. Cucker and S. Smale. The mathematics of emergence. *The Japanese Journal of Mathematics*, 2:197–227, 2007.
8. I. Licata and A. Sakaji, editors. *Physics of Emergence and Organization*. World Scientific, 2008.
9. E. Fehr and H. Gintis. Human nature and social cooperation. *Annual Review of Sociology*, 33(3):1–22, 2007.
10. R. Axtell. The emergence of firms in a population of agents: Local increasing returns, unstable nash equilibria, and power law size distributions. Technical Report CSED Working Paper #3, Brookings Institution, June 1999.
11. T. Van Belle and D. H. Ackley. Code factoring and the evolution of evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2002.
12. Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.
13. J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA, 1992.
14. W. Renz, T. Preisler, and J. Sudeikat. Mesoscopic stochastic models for validating self-organizing multi-agent systems. In *Proceedings of the 1st International Workshop on Evaluation for Self-Adaptive and Self-Organizing Systems*, Lyon, France, September 2012.
15. I. Fehervari and W. Elmenreich. Evolving neural network controllers for a team of self-organizing robots. *Journal of Robotics*, 2010.
16. A. Sobe, I. Fehérvári, and W. Elmenreich. FREVO: A tool for evolving and evaluating self-organizing systems. In *Proceedings of the 1st International Workshop on Evaluation for Self-Adaptive and Self-Organizing Systems*, Lyon, France, September 2012.