Evolving Self-organizing Cellular Automata based on Neural Network Genotypes

Wilfried Elmenreich and István Fehérvári

Mobile Systems Group, Lakeside Labs
Institute for Networked and Embedded Systems,
University of Klagenfurt

Abstract This paper depicts and evaluates an evolutionary design process for generating a complex self-organizing multicellular system based on Cellular Automata (CA). We extend the model of CA with a neural network that controls the cell behavior according to its internal state. The model is used to evolve an Artificial Neural Network controlling the cell behavior in a way a previously defined reference pattern emerges by interaction of the cells. Generating simple regular structures such as flags can be learned relatively easy, but for complex patterns such as for example paintings or photographs the output is only a rough approximation of the overall mean color scheme. The application of a genotypical template for all cells in the automaton greatly reduces the search space for the evolutionary algorithm, which makes the presented morphogenetic approach a promising and innovative method for overcoming the complexity limits of evolutionary design approaches.

Keywords: cellular automata, artificial neural networks, self-organizing systems, evolutionary algorithm

1 Introduction

The concept of Self-Organizing Systems (SOS), although long known from domains such as physics, chemistry and biology, has recently gained interest to be applied to technical systems. Self-organization can be defined as the emergence of coherent, global behavior out of the local interactions between components. This emergent organization is characterized by intrinsic autonomy, adaptability to environmental changes, and local awareness of the most important global variables. Most importantly, many SOS appear to be robust with respect to a variety of disturbances and intrusions, as the system is to some degree capable of overcoming or self-repairing damages. While many natural, social and technological examples of SOS exhibiting these characteristics are known, and several mechanisms of self-organization have been analyzed in detail, the design of a SOS remains a fundamental challenge [1]. Recent results have shown evolutionary design [2] to be a promising method for designing self-organizing systems [3,4].

However, as the system to be evolved becomes more complex, the evolutionary approach suffers from problems such as disruption of inheritance, premature convergence and failure to find a satisfying solution [5]. Yet, natural evolution has managed to create the very complex design of life. A main difference between natural and artificial evolution is, in many cases, the genotype-phenotype mapping. Natural organisms grow from

a single cell into a complex system, while in many applications of artificial evolution, there is a one-to-one mapping from genotype to phenotype, leading to poor scalability. Therefore, there is a strong need for introducing generic genotype descriptions that can emerge into arbitrarily complex systems.

In this paper we describe such an approach by the model of a cellular automaton where the state-transition logic of each cell is an instance of the same genotypical controller. The control algorithm is implemented by a small artificial neural network that is evolved to reproduce a given pattern on the cellular automaton. We tested the ability of the resulting pattern formation model for replicating several naturally occurring patterns (such as animal skin patterns) as well as man-made patterns (such as flags and paintings). Results show that the approach works better for regular structures.

The remaining parts of this paper are structured as follows: The following section 2 briefly reviews related work on pattern formation and artificial evolution approaches to evolve pattern formation processes. Section 3 introduces the model, i.e., the Cellular Automata (CA) structure and the properties and interconnections of the Artificial Neural Networks (ANNs). The evolutionary programming method is based on a tool named Frevo, its application for the given problem is described in Section 4. Experiments including evolving several patterns and discussion of the results is elaborated in Section 5. Finally, Section 6 concludes the paper and sketches possible further research and applications based on the presented approach.

2 Related Work

The need for proper genotype descriptions that can be evolved is discussed in the works of Bentley/Kumar [5], Eggenberger [6], and Miller [7]. Bongard and Pfeifer [8] demonstrate a model to evolve the morphology and neural control of an agent.

Amorphous computing [9] refers to systems of many identical simple processors each having limited computational ability and interacting locally. Typically, such systems are of irregular structure in contrast to the regular structure of CAs as they are used in our case study.

Pattern formation in general is studied in developmental biology in terms of cell fate control through a morphogen gradient. A morphogen has been conceptually defined in the 1960s by Lewis Wolpert using the French Flag Model [10] where the colors of the French flag represent the effects of the morphgen on cell differentiation. Herman and Liu [11] have solved the French flag problem through the simulation of linear iterative arrays of cells. Miller [7] has created a simulated differentiated multicellular organism that resembles structure and coloring of the French flag. He uses a feed-forward Boolean circuit implementing a cell program. The cell programs are evolved using a specialized Genetic Programming system. Miller also comments on the difficulty of the problem of evolving a cell program.

Chavoya and Duthen in [12] have used a Genetic Algorithm to evolve Cellular Automata that produce 2D and 3D shapes such as squares, diamonds, triangles, and circles. Furthermore, in [13], they have evolved an Artificial Regulatory Network (ARN) for cell pattern generation, producing the French flag pattern.

Fontana in [14] generated predefined arbitrarily shaped 2D arrays of cells through an evolutionary-developmental technique. He was able to successfully generate complex

patterns such as dolphin, hand, horse, foot, frog, and the French flag. Fontana has extended this work in [15] to evolve complex 3D forms.

The combination of neural networks and cellular automata which grow under evolutionary control is present in the China-Brain project [16], an effort to create an artificial brain of interacting small (typically 12-20 neurons) neural networks. This work, having several aspects in common, differs from the work presented in this paper in two aspects: China-Brain is intended as a controller (e.g., to a robot) while in our problem the structure and form of the cells themselves are the output and, second, the interconnections between the modules are designed explicitly by so-called brain architects, while they are an output of the evolutionary process for our model.

3 Cellular Automaton Model

The used model consists of a regular rectangular grid matching exactly the resolution and proportion of the reference image (reference images are very small, typically 50-500 pixels). The colors of the reference image are converted into a scale, where neighboring colors are resembling each other. This color transformation from RGB has been achieved by assembling a binary number by arranging the most significant bits of the channels R,G and B, followed by the second most significant bits of those channels, and so on until the least significant bits, finally yielding a 24-bit color code.

The colors which are present in the reference image are then sorted according to the new scale and assigned to the possible output spectrum of the neural networks. Each color gets an equal proportion of the output space, which is continuous between -1 to 1.

Each cell is controlled by an ANN, which is modeled as a time-discrete, recurrent artificial neural network. Each neuron is connected to every other neuron and itself via several input connectors. Each connection is assigned a weight and each neuron is assigned a bias value.

At each step, each neuron i builds the sum over its bias b_i and its incoming connection weights $w_j i$ multiplied by the current outputs of the neurons $j = 1, 2, ..., n$ feeding the connections. Weights can be any real number, thus have either an excitatory or inhibitory effect. The output of the neuron for step $k + 1$ is calculated by applying an activation function F:

$$o_i(k+1) = F(\sum_{j=0}^{n} w_{ji} o_j(k) + b_i)$$

where F is a simple linear threshold function

$$F(x) = \begin{cases} -1.0 \text{ if } x \leq -1.0 \\ x \quad \text{if } -1.0 < x < 1.0 \\ 1.0 \quad \text{if } x \geq 1.0 \end{cases}$$

In total each ANN consists of 9 input neurons, 5 output neurons and 6 hidden neurons. The 6 hidden neurons have been selected after a short evaluation and turned out to be an applicable number in order to balance between capability of the ANN and reduction of the search space. One outgoing connection is used to define the cell's color and four pairs of incoming and outgoing neurons connect the ANN with neighboring cells.

Furthermore, an ANN can sense the colors of the neighboring cells. The ANN of a cell does not get any direct information about its position in the grid. A cell at a border or in corner, however, would be able to infer about its position. Via the inter-cell connections, information can propagate to the cells in the center.

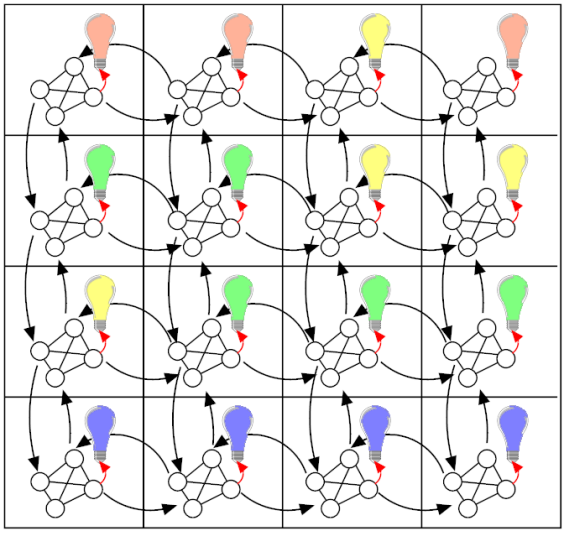

Figure 1. Interconnections of ANNs in neighboring cells

Figure 1 depicts the interconnections between the ANNs via neighboring cells in the CA model. The light bulb, which is controlled by the ANN in the same cell indicates the current color state of a cell. Each cell gets an instantiation of the same ANN. When the CA is iterated, its ANNs can however acquire a different internal state that can be kept via self-holding loops and that can lead to differentiation in the ANN's behavior.

4 Evolutionary Programming of the ANNs

In order to evolve the weights of the ANN, we used Frevo [3], a Java framework for generating distributed self-organizing systems.[1] Frevo allows to combine different *representations* (e.g., fully-connected ANNs, layered ANNs, Finite State Machines (FSMs)) with an *optimization method* and a *problem* to be solved. Every problem contains a generic interface to one or several instances of the representation, the optimizer basically evolves a control algorithm formulated in the representation that solves the problem.

The modular concept allowed to reuse the model for the ANN and the optimization algorithm from previous projects, thus reducing our task to formulate and implemented

[1] Frevo and most of its models are available as open source software at `http://www.frevotool.tk/`

the problem. The used optimization algorithm is depicted in the algorithm below. The selection criteria are based on the rank (according to the candidate's fitness) and, in case of the random selection, also on diversity. Diversity means that candidates which are more different to the already selected pool of candidates have a higher chance to be chosen. Functions for mutation, recombination and the difference between candidates are provided by the specific candidate implementation. In our case, the ANNs applied mutation and recombination on the weights and biases of the artificial neurons. The difference between the candidate is a sum of the squared differences of all weights and biases. The number of neurons and structure of the ANNs was fixed at runtime, since the problem statement has no dynamic aspects and networks with fixed structure have shorter evolution times [17].

Algorithm 1 Evolutionary algorithm used as optimization method

```
1:  create n networks in a population and initialize them with random values
2:
3:  for generations
4:     for i=0 to n
5:        evaluate network_p, i and store score
6:     rank networks according to their score (best first)
7:     select elitist networks
8:     select randomly networks (bias for better ranked and diverse networks)
9:     create mutations of selected networks
10:    create recombinations of selected networks
11:    create some networks anew and initialize them with random values
```

The parameters used for the the evolutionary algorithm are listed in Table 1.

Table 1. Parameters used for the evolutionary algorithm

Population size	100
Elite selection	15%
Random selection	10%
Mutated networks	30%
Recombination	40%
New networks	5%
Mutation rate	5%

A problem consists typically of a simulation with a generic interface to the control system. The simulation returns a fitness value which is used by the optimization algorithm for evolving the system. We implemented a CA simulator that is controlled by a representation. The fitness function was implemented as a sum of the squared color index differences between the image after a number of iterations and the reference image. In order to approximate human perception of different images (human vision is focused around edges rather than areas with the same or similar color), the summands have been

weighted by a function giving higher values for pixels having pixels of different color in their neighborhood.

5 Experiments and Results

We found flags to be the simplest kind of patterns to emerge for our proposed set-up. Figure 2 depicts the evolution of a Hungarian flag. The reference image consists of a 6x9 image, where the two top rows are red and the top lower rows are green with white in between. Thus the complexity of this reference is equivalent the French flag which was used as reference in several related work articles as described in Section 2.

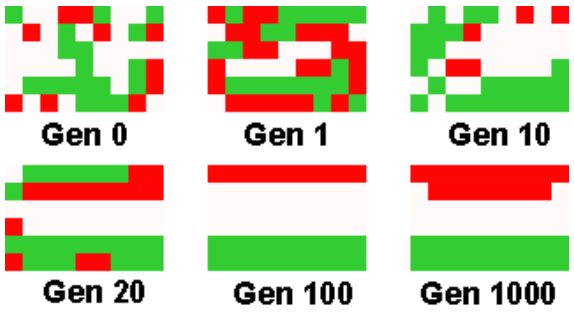

Figure 2. Evolution of recreating a Hungarian flag

Note that the proceedings in the quality were highly non-linear over the number of generations, because the evolutionary algorithm gets often stuck in local cost minima after about 100 generations. Thus, improvements past these generations happen only very infrequently.

The mechanism to recreate an image over several iterations of the CA can be observed by the example of an Austrian flag. The Austrian flag contains only red and white color and was therefore easier to evolve than the three-colored Hungarian one. We achieved a perfect reproduction after running the evolutionary algorithm for 90 generations. Figure 3 shows how the result unfolds over several CA iterations into the intended image.

The limits of the approach can be observed when going to more complex images. Figure 4 depicts the results of trying to reproduce a small image of the Mona Lisa painting (left image). The middle image shows the downsized reference image. The best achievable result after over 500 generations is depicted in the right image. The overall background color scheme is present, although, unfortunately, Mona is missing. The main reason for this result lies in the increased size of the image – while the flags were evolved on a raster of 6x9, the Mona Lisa image is 20x29. Note that initially only cells at the corner and the borders can detect their position in the image - the inner cells must rely on propagating information. For a larger image, the ratio between border and inner cells is more extreme.

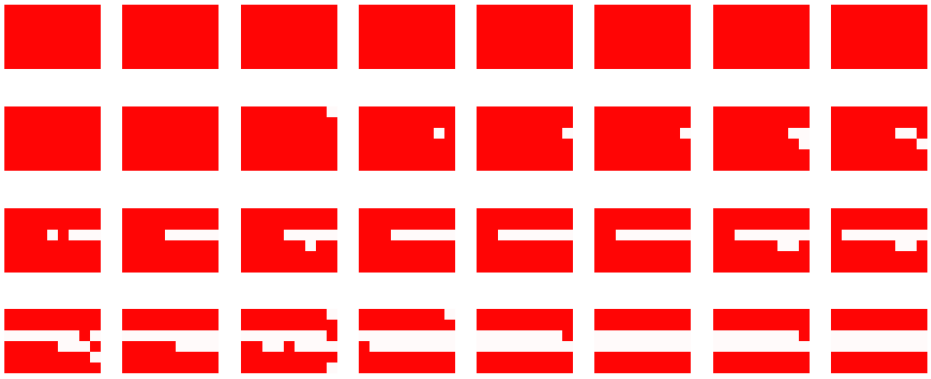

Figure 3. CA steps for Austrian flag

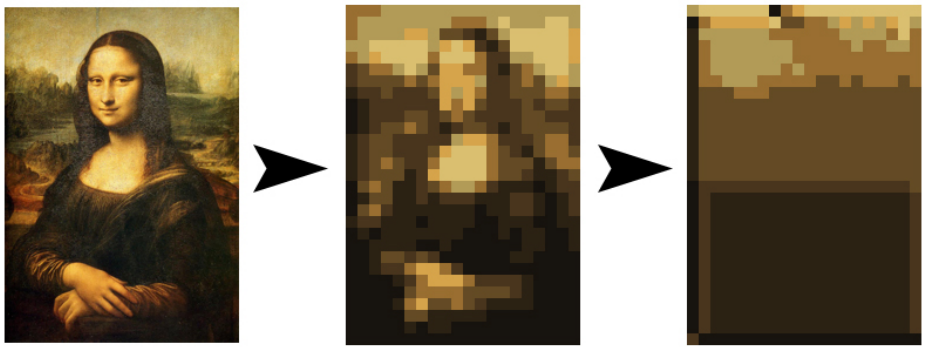

Figure 4. Attempt to reproduce the work of Leonardo da Vinci

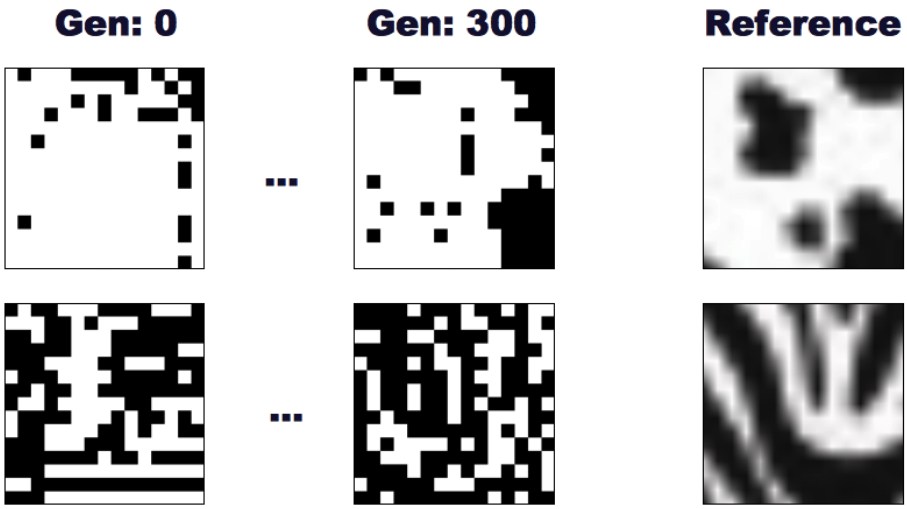

Figure 5. Evolving a reproduction of animal skin patterns

Another question of interest was how well natural patterns can be evolved. Several patterns resembling natural ones can be reproduced by CA executing simple state-transition rules of positive and negative feedback [18].

Interestingly, our evolutionary algorithm did not come up with a feasible solution. This is likely because the fitness function was inappropriate for that task, since it compared the potential solution pixel by pixel to a reference image. Thus, a *similar pattern* is not considered as solution, although a human observer might perceive a similar pattern as being closer to the reference than an image that partially reproduces the original layout of objects in the image. Figure 5 shows that although the created image resembles the reference one on a pixel-by-pixel basis, the quality and type of the reference pattern is not matched.

6 Conclusion and Future Work

We depicted and evaluated a design process that generates a multicellular system out of a genotypical description for a single cell. The mechanisms have been realized via an open source framework for evolutionary design (FREVO). At the beginning of each simulation, all cells had the same state and commenced their operation at the same time - this is comparable with a number of people cooperatively drawing an image in the dark. This differentiates our problem from the ones in the literature, where usually a zygote cell is given, from where the other cells grow. Still, the evolutionary process evolved a solution where also some eminent cell (typically a particular corner cell) serves as a zygote.

The main contribution of this paper is not presenting an algorithm for "drawing images in the dark" but rather presenting a proof-of-concept on integrating ANNs into a CA in order to initiate a morphogenetic process.

Possible applications of this research could be the self-organized pattern formation in swarm robotics. In other words, given a desired pattern, how can robots acquire it? Another application could be smart paint (as indicated in [9]) that would decide on its color based on a morphogenetic process having only a few distinctive sensory inputs, thus not allowing for a zygote approach.

The best results have been achieved when evolving simple structures with large areas of a single color as they are present for example in flags. For more complex images, the current setup causes the evolutionary algorithm to get stuck at a suboptimal stage. There is, however, a large space of possibilities for variations of the model which gives rise to future work. E. g., findings on well-suited or less well-suited model configurations could give insight to the understanding of such phenomena as morphogenesis and camouflage mechanisms in nature.

Future experiments are planned to involve modifications in the internal ANN structure (e.g., investigate on the optimal number of hidden nodes for different reference images) as well as increasing or decreasing the ability of ANNs to communicate with their neighbors. For evolving structures rather than replications of images, we are planning to design the fitness function in a way to have the fitness based on the type of the emerging structure instead of a pixel-by-pixel comparison.

Acknowledgments

This work was supported by the European Regional Development Fund and the Carinthian Economic Promotion Fund (contract KWF 20214|18128|26673) within the Lakeside Labs project DEMESOS and the follow-up project MESON. We would like to thank Marcin Pilat, Miguel Gonzalez, and Rajesh Krishnan for their input on earlier versions of the paper. Furthermore, we would like to thank the anonymous reviewers for their constructive comments.

References

1. W. Elmenreich and G. Friedrich. How to design self-organizing systems. In *Science beyond Fiction FET09*, pages 61–62, Prague, Czech Republic, 2009. European Commission: Information Society and Media, Brussels.
2. P.J. Bentley. *Evolutionary Design by Computers*. Morgan Kaufman, 1999.
3. I. Fehérvári and W. Elmenreich. Evolutionary methods in self-organizing system design. In *Proceedings of the 2009 International Conference on Genetic and Evolutionary Methods*, 2009.
4. I. Fehervari and W. Elmenreich. Evolving neural network controllers for a team of self-organizing robots. *Journal of Robotics*, 2010:10 pages, 2010.
5. P. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *In Proceedings of the Genetic and Evolutionary Computation Conference*, pages 35–43. Morgan Kaufmann, 1999.

6. P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the Fourth European Conf. Artificial Life (ECAL'97)*, pages 205–213, 1997.
7. J. F. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2004)*, pages 129–139, 2004.
8. J. C. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 829–836, San Francisco, CA, USA, 2001.
9. Harold Abelson, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F Knight, Radhika Nagpal, Erik Rauch, Gerald Jay Sussman, and Ron Weiss. Amorphous Computing. *Communications of the ACM*, 43(5):74–82, 2000.
10. L. Wolpert. Positional information and the spatial pattern of cellular differentiation. *Journal of Theoretical Biology*, 25:1–47, 1969.
11. G. T. Herman and W. H. Liu. The daughter of celia, the french flag and the firing squad. In *WSC '73: Proceedings of the 6th conference on Winter simulation*, page 870, New York, NY, USA, 1973. ACM.
12. A. Chavoya and Y. Duthen. Using a genetic algorithm to evolve cellular automata for 2d/3d computational development. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 231–232, New York, NY, USA, 2006. ACM.
13. A. Chavoya and Y. Duthen. Use of a genetic algorithm to evolve an extended artificial regulatory network for cell pattern generation. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1062–1062, New York, NY, USA, 2007. ACM.
14. A. Fontana. Epigenetic tracking, a method to generate arbitrary shapes by using evolutionary-developmental techniques, May 2008.
15. A. Fontana. Epigenetic tracking: A possible solution for evo-devo morphogenesis? In *Proceedings of the 1st International Workshop on Morphogenetic Engineering*, 2009.
16. Hugo de Garis, Jian Yu Tang, Zhiyong Huang, Lu Bai, Cong Chen, Shuo Chen, Junfei Guo, Xianjin Tan, Hao Tian, Xiaohan Tian, Xianjian Wu, Ye Xiong, Xiangqian Yu, and Di Huang. The china-brain project: Building china's artificial brain using an evolved neural net module approach. In *Proceeding of the 2008 conference on Artificial General Intelligence 2008*, pages 107–121, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.
17. Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.
18. Y. Bar-Yam. *Dynamics of Complex Systems*. Perseus Books, 1999.