Cellular Automata with Irregular Structure: a Compact Representation

Jana Baran, Pavel Petrovič, and Marc Schoenauer

***Abstract*— Cellular Automata (CA) are a standard theoretical model of uniform parallel computation on a grid of cells. They can be looked upon as a discrete type of dynamic systems. As such, they are important tools for modeling spatially distributed processes of different kind – from ecology, through biology, and artificial life, to economics. They are also a useful theoretical model for studying classes of computational complexity. We are interested in the ability of CA to converge to a fixed point with interesting properties, which are quantized by an objective function. In this way, CA qualify as a promising embryogenic representation for Evolutionary Design. Moreover, by modifying the original CA concept by allowing irregular mesh, we can achieve a more flexible and compact representation resulting in faster evolutionary progress.**

***Index Terms*—cellular automata, evolutionary design, evolutionary algorithms, irregular grid.**

I. Introduction

CELLULAR Automata are a theoretical model consisting of a set of cells, typically arranged in a form of a grid, or array. Each cell has a set of possible states it can enter, and each cell has a defined neighborhood – a set of adjacent cells. The next state of a cell is determined by the current state of the cell and states of the neighboring cells. Rules that specify the state transitions are common to all cells in the automaton. CA have been intensely studied by various researchers. For instance, Stephen Wolfram in his book New Kind of Science studied the evolution of one-dimensional CA in time, and examined its regular, fractal and chaotic behavior [1]. A famous example is the Game of Life of John H. Conway, a two-dimensional CA, which was shown to be a universal computer, capable of computing an arbitrary algorithm [2]. Actual implementations of universal computers using Life have been constructed more recently, see [13] for a list. Other types of CA, including hexagonal and reversible CA – i.e. those able to compute in reverse direction back to recover their inputs – have also been shown to be universal computers [14]. However, in most studies, the basic property of CA has not been relaxed – namely that they are formed of cells of equal sizes, arranged in regular grids or meshes, with equal neighborhood structure (except of the border cells). We believe that even though such CA might be more amenable to theoretical analysis, useful practical results may be obtained with automata with varying cell size, irregular grid structure, and varying neighborhood shape. Such CA may save large resources by substituting large areas that are filled by many regularly arranged cells by a single cell that is capable of performing the same functionality. We demonstrate this idea on a case study from Evolutionary Design, where the goal in general is to design a target shape (2D or 3D) that satisfies required criteria. We will show how irregular CA outperform as a representation type the standard CA with regular grid. In the following sections, we make a few notes about related work with irregular CA, introduce Evolutionary Design, the evolutionary algorithm we use, different representation types, CA as embroygenic representation, our example task, our proposed irregular CA, results we obtained, and finally add concluding remarks and ideas for the future work.

Manuscript received July 12, 2010.

J. Baran and P. Petrovič are with the Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Mlynská dolina, 842 48 Bratislava, Slovakia, janahlava@gmail.com, ppetrovic@acm.org.

M. Schoenauer is with Laboratoire de Recherche en Informatique, Université Paris Sud, 914505 Orsay Cedex, France, Marc.Schoenauer@lri.fr.

II. Related Work

The inspiration for somewhat more dynamic structure of cells came from J. F. Miller who introduced Cartesian Genetic Programming in the French flag problem [7]. He used cellular representations and an updating engine. In his model, program of each cell decides on the amount of produced chemical, whether it will live, die, or change to a different cell type at the next time step, and how it will grow. Growing into another cell means overwriting its properties completely. Besides the three color states, there is also another state – dead. When a cell dies, it means that it does not act any more. Even though the cells still live and grow on locations placed on a regular grid, Miller's work has inspired us to do experiments with a dynamic grid of cells.

H. de Garis used in his work [4] two dimensional shapes formed by a colony of cells in reproductive CA as embryos. The idea was to evolve reproduction rules for CA, such that the final shape of a colony of cells would match a desired shape as closely as possible. Each cell contains a differentiable chromosome, which consists of four 'operons'. Each operon contains a condition field and an action field. These operons can switch on and off over time. The sequential operon switching controls the growth of an embryo. If a cell matches one of the conditions of an operon, then the corresponding action is activated and its instructions executed. The matching is computed from the state of the cell, which is computed from the previous cell state and from states of the neighbors of the cell. The state of a cell was defined in terms of the configuration of its neighbor-less side(s) because only such cells can reproduce (there are 14 different states). Both these works showed some capability of a dynamic irregular cells structure to learn a desired shape, i.e. optimize the target pattern.

III. Evolutionary Design

Evolutionary design is one of the application domains of evolutionary computation, it extends beyond Computer-Aided Design (CAD), and it borrows ideas from natural evolution. Evolutionary design has been applied in many different areas over the last decades, in mechanical engineering to optimize structures (flywheels, propellers, wind turbines, supersonic aircrafts, etc.), in electrical engineering to optimize circuits, or in computer engineering to optimize hard problems (for example non-polynomial problems) [3, 4]. Even more, evolutionary design was used as not only the optimizer but as a creator, especially in different kinds of art, modern architecture, or in computer science to evolve artificial life [3, 5]. In general, the types of evolutionary design can be divided into four main categories: evolutionary design optimization, creative evolutionary design, evolutionary art, and evolutionary artificial life forms. For each Evolutionary Design application, three most important implementation challenges must be faced: first, the type of an evolutionary algorithm, which is responsible for the organization of initializing, selecting, recombining, and mutating the potential solutions, second, the representation type, i.e. how to uniformly encode all potential solutions, and finally, the palette of the evolutionary operators – the ways how the mutation and recombination change the selected individuals. An additional aspect is setting of the parameters, which is usually based on empirical and some theoretical assessment. We describe the important choices in the following sections.

IV. Evolutionary Algorithm

For the evolutionary algorithm we chose the state of the art method of Nikolaus Hansen, Covariance Matrix Adaptation Evolution Strategy (CMA ES) [6]. Evolutionary strategies as contrasted to Genetic Algorithms rely more on mutation than random recombination of the crossover type. The populations tend to be smaller, and they are not represented by a set of individuals, rather, they are represented by a probabilistic distribution of multivariate normal distribution. The individuals are sampled from this distribution at the beginning of each generation. At the end of each generation, the individuals adjust the parameters of the distribution according to their performance and then they die. This kind of algorithm is typically very useful in searching for vectors of real-valued parameters in a smooth fitness landscape. However, various additional techniques, such as evolution path and step-size control make this method a successful one, even in problems with multi-modal, ill-conditioned, and non-separable fitness landscapes. All details can be found in [6].

V. Representation Types

A crucial step in the design of an evolutionary solver is the selection of an effective representation for the current problem. The very important consideration is that if two individuals are closely related on the genotype level then they should be closely related on the phenotype level as well. Otherwise, two solutions are incomparable. In other words, a minor change in the genotype should not cause a major change in the phenotype, otherwise the fine tuning of the system becomes difficult. Another important issue to consider before choosing an appropriate representation/embryogeny is the dimensionality of the search space and the level of complexity. An efficient representation/embryogeny can provide the following benefits: reduction of the search space, complex phenotype solutions, constraint handling, adaptation and repetition. The advantages and drawbacks will be further discussed for each representation category. Representations versus embryogenies:

- direct representations (no or external embryogeny)
- indirect or generative representations (explicit or implicit embryogeny)
- cellular representations (implicit embryogeny)

Direct representations are the simplest type of representations where the genotype directly encodes the phenotype. The simplest direct representation is a binary representation, so called bitarray [8], where each bit represents a single unit of the design pattern, a pixel or voxel (Figure 1). Examples of evolutionary operators in this case would be random initialization, simple bit-mutation, and some kind of geometric crossover: combining coherent components of two parent solutions to obtain offspring.

Fig. 1. Example of a direct representation.

More complex and sophisticated representations are *indirect representations*. In indirect representations, the phenotype is determined by the genotype, but the genotype does not encode the phenotype directly, but rather encodes instructions how to construct the resulting phenotype. An example of generative representations are L-systems, see for example [9], unordered lists, program trees, or graphs. More specific examples include Voronoi diagrams, dipoles representations, and Iterated Function System (IFS) [10]. In this case, the evolutionary operators are typically working on a genotype encoded for instance by a set of vertices – seeds for Voronoi diagram (Figure 2). Mutating thus moving the seeds leads to altering the resulting shape correspondingly.

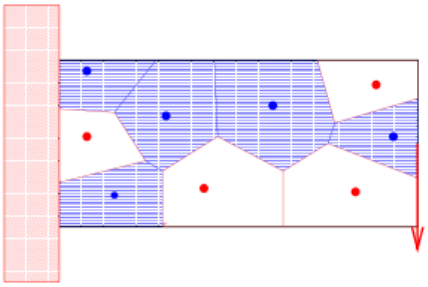

Fig. 2. Example of an indirect representation: Voronoi representation for the 2x1 cantilever test problem [15].

Cellular representations are a very promising representation type used in evolutionary design. They combine the properties of direct and indirect representations in the sense that the cellular topology (for instance a grid) maps to the phenotype topology directly. For instance, the cellular grid can map one to one with the resulting picture, thus a cell can match a pixel. On the other hand, the color or other properties of the cells are gained implicitly thus indirectly (Figure 3).

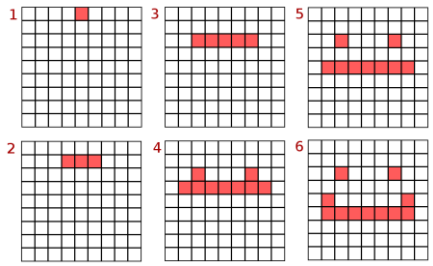

Fig. 3. An illustration of a cellular representation, the resulting shape is the outcome after several iterations of the CA.

In reference to the natural inspiration to Evolutionary Algorithms, generative representations are often extended by an embryogenesis – a natural process of development from embryo to fetus, or simply a process of growth that defines how a genotype is mapped onto a phenotype. In the terms of CA, the transition function of a single CA cell is a genotype, and the constructed CA is an embryo that develops to a final target shape – a phenotype through a series of iterations.

VI. Example Task

For the purpose of this work, we consider a task of constructing a 2D shape with an exact prescribed pattern. As such, this task is not really a practical application of Evolutionary Design, but rather a testing of the representation. We would like to see our algorithm constructing novel and unanticipated shapes that would, however, show the expected qualities and properties. Obviously, that is the ultimate goal of the engineers, but we, trying to stand on the science side, are more interested in studying the properties – namely the evolvability and performance of the different representations. Therefore, the exact purpose of the objective function is secondary to our interest, and comparing to a specific output pattern serves well to see how flexible a proposed representation could be.

Our starting point was a dissertation of Alexandre Devert [11], who successfully evolved CA iterating to a stable fix-point – a target shape in form of a 2D pattern – simply called a "flag" (Figure 4). In particular, by means of the CMA-ES evolutionary algorithm, Devert optimized weights of a feed-forward neural network (NN) that controlled the state transition function of the cells. Inspired by the Turing's diffusion-reaction system, Devert's CA also uses diffusion of chemicals, i.e. smoothing the spreading of states through the automaton using the application of Gaussian blur operator. Note that CA usually contain cells that take upon a discrete set of states, while in this case, a cell state is a vector of arbitrary real values. This makes the states less crisp, more continuous and thus more amenable for evolution and smooth behavior of artificial NN. Using a vector of numbers allows storing independent values, for instance corresponding to distances from left and top borders. However their actual meaning hidden from us, it is discovered by the evolutionary algorithm. Figure 5 shows a diagram explaining the states of a cell. The states are of two types: internal states and external chemicals. The transition function is a standard multi-layer perceptron (MLP) with a fixed number of hidden nodes. Thus the genotype in this case consists of a vector of input weights for all hidden and output neurons, which are directly connected to the new cell states and chemicals. In more detail: the input for the MLP is the chemical vector consisting of the chemicals of the current cell with chemicals of its neighbors, and the state vector consisting of the states of the current cell. The output of the MLP are the new values for states, chemicals, and optionally the cell merge/split control signal or the cell color signal – determining the resulting pattern. The inputs to the hidden layer are the states, and neighbor chemicals but not the chemicals of the current updating cell. All input values are connected to all hidden neurons. Similarly, all hidden neurons are connected to all output values. Output values consist of new states and chemicals of the cell. Furthermore, the layer of neurons computing the output chemical values has another input also - the chemicals of the current updating cell that were omitted for the hidden layer (Figure 5). The output value for each of the hidden neurons and perceptron final outputs are computed by applying the activation function - hyperbolic tangent of the sum of all its weighted inputs together with the previous neuron value. The colors of all the cells are computed as:

$$color_i = \frac{1 + \tanh\left(w_{c_i} c_i + w_{s_i} s_i\right)}{2} \quad (1)$$

where i is the i-th cell, c_i is its chemical vector, s_i is its state vector, w refer to weights.

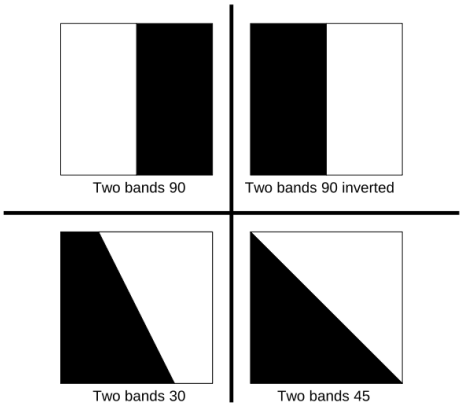

Fig. 4. Examples of target patterns (flags) used in our experiments.

The objective function simply compares the specified and resulting patterns, for both regular and irregular CA:

$$d = \frac{\sum_{i=1}^{cells} \left(colorCell_i - colorPixel_{random}\right)^2 * size_i}{size_{max} * cells} \quad (2)$$

where *cells* is a number of cells in the embryo, $colorCell_i$ is a color value of the cell, $colorPixel_i$ is a random point in the picture covered by the cell, $size_{max}$ is the maximum size of a cell in the embryo.

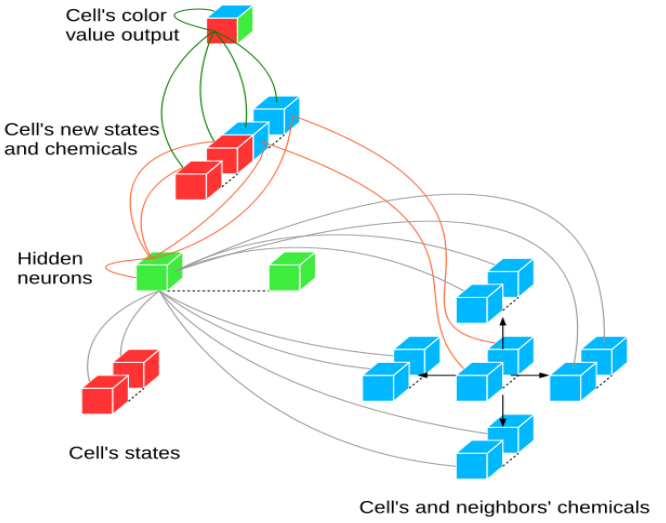

Fig. 5. Cell state transition is guided by a neural network. States are of two types: internal states and external chemicals that diffuse and interact with neighboring cells [5].

In our previous work [12], we have successfully repeated Devert's experiments (Figure 6) and modified the algorithm in such a way, that the same CA was able to converge to two different flags (fix-points) depending on the environmental conditions – i.e. fixed inputs provided to the cells on the border of the CA. Automaton was always initialized with zeros as initial states in all cells, but the cells on the border were connected to "constant" cells that were either 0 or 1 – depending whether the automaton was to produce the first or the second required pattern.

Fig. 6. Examples of resulting patterns of CA controlled by an evolved transition function [5]. A reconstructed experiment of Devert [11].

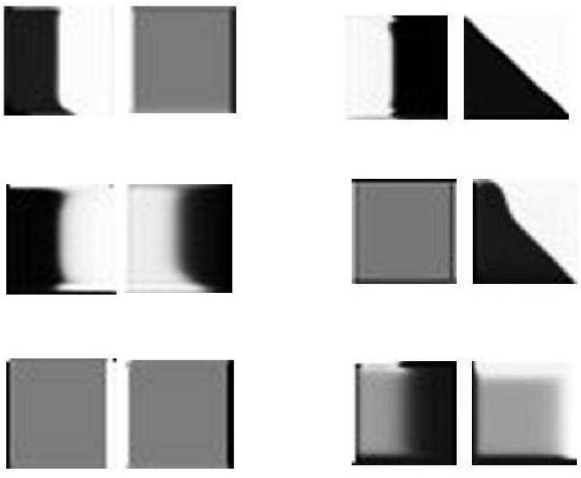

Fig. 7. Example of two flag patterns represented by the same CA: on the left, two bands 90 and two bands 90 inverted, on the right, two bands 90 and two bands 45. All possible outcomes – no fix-points, one fix-point and two fix-points are shown.

Figure 6 contains example outcomes of a single fix-point CA. Figure 7 shows two evolved shapes that are represented by a single CA able to reach two fix-points (attractors). Our attempts to evolve more than two fix-points by varying the content of the constant border cells did not find a solution. However, we would like to try different methods, for instance providing a different constant input to all the cells in the CA.

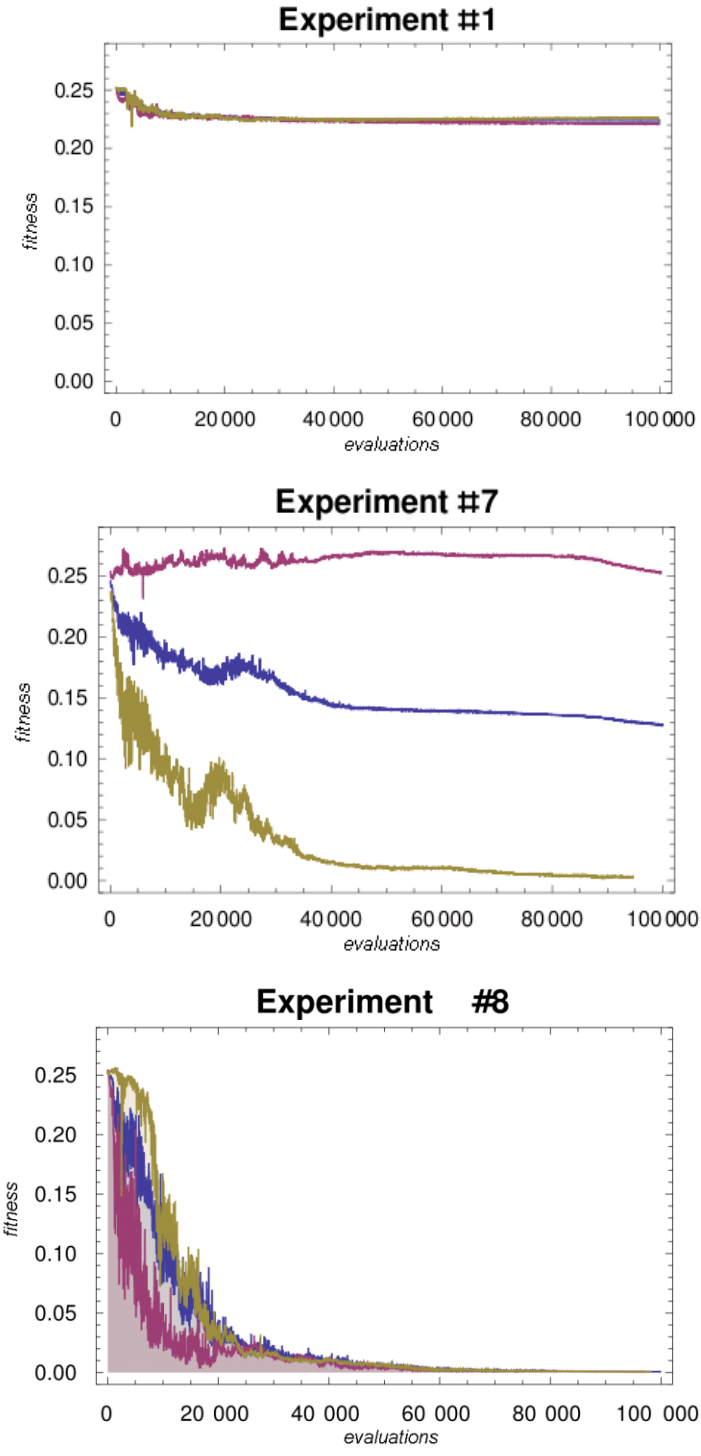

Fig. 8. Plot of three different evolutionary runs showing the fitness of both shapes (and their average) against the number of evaluations. The first case was unsuccessful, second case evolved only one fix-point, while the third case evolved both fix-points.

In this work, returning back to our discussion in the introduction, we are looking at the possibility to deal with CA that do not have a regular grid structure.

VII. Irregular CA Structure

A newer version of our CA consists of irregular rectangular grid of cells that can be of different sizes. The color of a specific cell is a part of the target pattern in the same way, regardless its size. Our goal is to let the CA decide the structure of the cells on its own, based on the pattern. Thus the cell states determine not only the resulting color but also whether the cell wants to merge with neighboring cells or split. If the signal is positive (negative) and above (below) some threshold, it will try to merge (split), and there is a neutral interval around zero when the cell remains in the same size. Cells are allowed to merge only in a square manner – i.e. when any four cells that form a square together like to merge, they are all replaced by a single cell. The resulting cell states are determined as an average of the merged cells. A cell willing to split is allowed to do so at any time, and the four newly born cells are arranged in a square, receiving copies of all the parent cell states.

At the same time, we would like to find a suitable transition function (i.e. MLP in our case), which we could install in all the cells, again, regardless of their size, to obtain a resulting pattern with the required properties. Experiments persuaded us that trying to do these two things simultaneously (optimize structure and color) given our representation is not necessarily a good idea, and thus we tried the incremental approach: first find a suitable structure and only then find a good transition function once the structure is fixed, although irregular. We can hypothesize that simultaneous evolution is difficult, because changing the topology, while the transition is being searched for is too destructive. Finding transition is challenging already when the topology is static, but making it dynamic complicates the matters strongly.

Optimizing the structure of a picture means adaptation of cell sizes and positions to the pattern of the picture. The pattern of the picture consists of contours in the picture, the borders of monochromatic regions in the picture. In the optimal case, the cells would adapt to the various color regions of the picture, and each cell would match a different region at the end. However, our grid is not so flexible as it is allowed to consist of square cells only. The shape of color regions is simply approximated using the squares with sides of length 2^k. At the beginning, each cell corresponds to one pixel of a target picture. The cells are allowed to do one of the three actions: merge together, split, or do nothing. The desired growing action of a cell is determined by merge/split signal computed by the MLP controller. The objective function in this case penalizes all cells that cover more than one color region. In particular, the numbers of pixels in all color regions, except the largest one that the cell covers, are summed together for all the cells in the automaton. Secondly, the objective function penalizes the number of cells used, i.e. the algorithm will try to minimize the number of cells. The balance between the two criteria is determined by the size of the grid, but there is a space for improvement. Alternately, an approach using a multi-objective evolutionary algorithm could be used to get a balance between the two criteria. The objective function we used is as follows:

$$d = \frac{\sum_{i=1}^{cells} min\left(black_i, white_i\right) + \frac{cells}{width}}{min\left(\sum_{i=1}^{cells} black, \sum_{i=1}^{cells} white\right) + height} \quad (3)$$

where *width*, *height* are dimensions of the embryo, $black_i$, $white_i$ is the number of black/white pixels respectively per each cell, *black*, *white* is the total number of black/white pixels respectively in the embryo.

Once the topology was optimized, i.e. the best fitness in the population stopped to improve, the second stage of color optimization automatically followed. It used the same objective function as in the regular CA, (2).

VIII. Results

We performed the first experiments with only the first stage of the evolution – to see if we can optimize the structure of the grid at all. They showed the capability of the CA to adapt its structure to the structure of the target pattern. Some examples of resulting optimized structures of the 33x33 pixels pictures are shown in Figure 9, the shading shows the underlying color pattern.

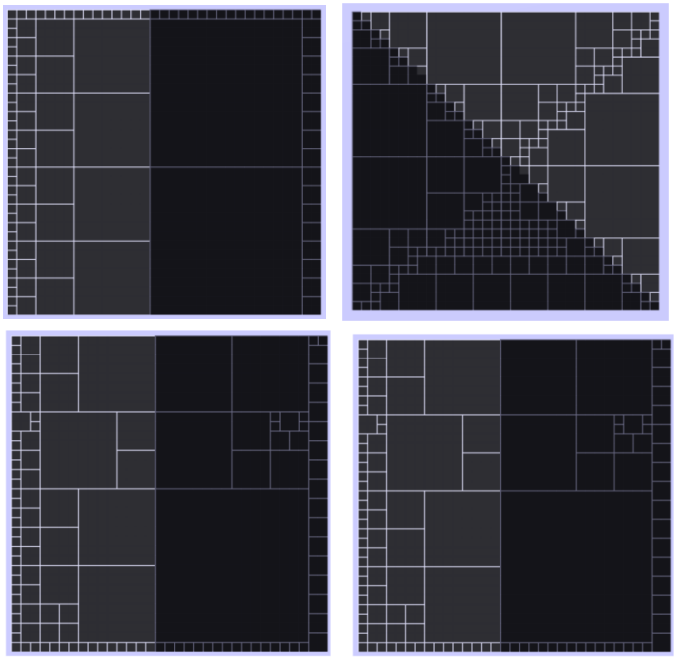

Fig. 9. Examples of resulting optimized patterns (structure).

The success in structure optimization was a necessary condition for the optimization of both the structure and the color. Evaluations of average best fitness values of experiments performed to optimize the structure and then the color of the targets are shown in Figure 10. Examples of the resulting optimized structures with colors of the 33x33 pixels pictures are shown in Figure 11. The total number of evaluations was shorter in case of the staged evolution with evolving the structure first than in case of regular grid CA, although the color mapping was not so good. Further experiments might be needed to investigate the potential.

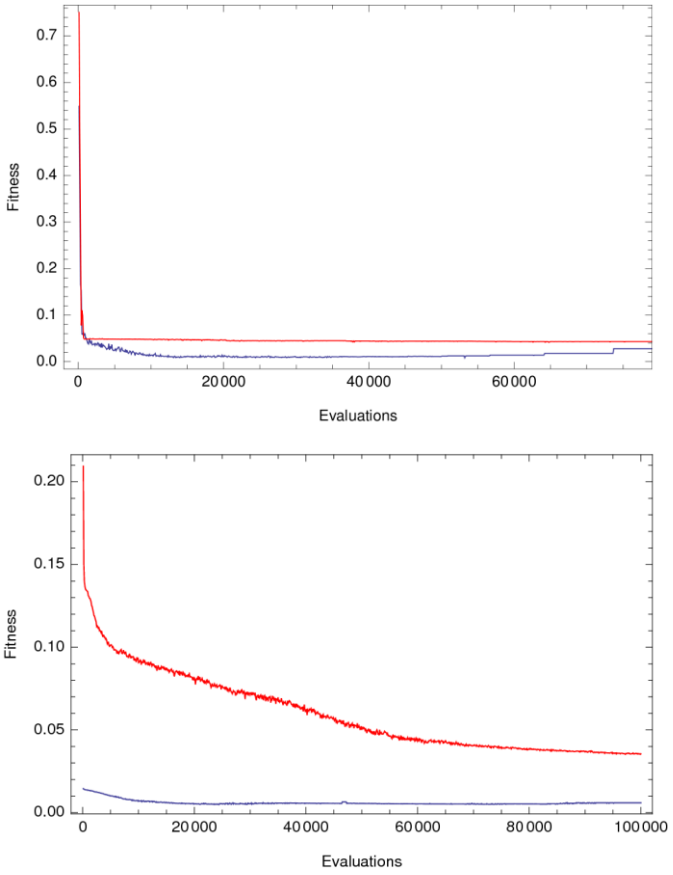

Fig. 10. The average best fitness values achieved by optimization of the structure (top) and then color (bottom of two patterns (two bands 45 – converges more slowly, and two bands 90 – converges faster).

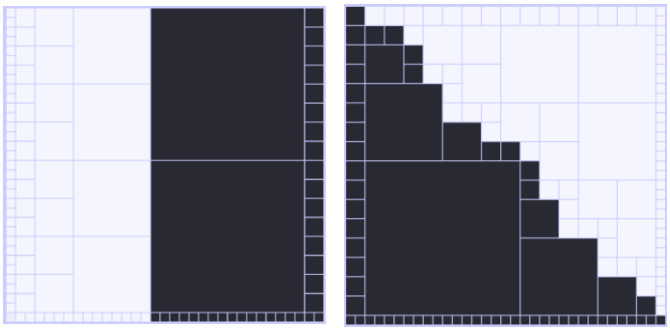

Fig. 11. Examples of resulting optimized patterns (structure and color).

IX. Conclusions and Future Work

In this case study of Evolutionary Design, we investigate Cellular Automata as embryogenic representation type. We build upon a successful work [11] that used regular-grid CA as a representation to construct a specified 2D color pattern. The work was later extended to find CA that can represent two different flags. Our interest is in modifying the regular grid structure of the automaton in order to provide a more flexible representation that distributes the resources based on the local complexity in different parts of the pattern. We show that using CA with irregular grids and cells of different sizes produces reasonable patterns, with more compact representation of the resulting shape. We also achieved this faster than using a regular-grid CA. In the future work, we would like to study the CA with irregular mesh in more details.

In our work, we tried only one specific type of cell merging/splitting, while various different types are possible. For instance, one could replace the first stage of the evolution with a deterministic method – for instance rendering an optimal quad-tree that fills the color regions without conflicts, or allowing cells of non-square shapes. It remains yet to try, if irregular automata can also evolve multi-fixed-point CA as we showed for the regular ones. We are also interested to learn about other work that relates to CA with irregular grid, in order to improve our knowledge on developmental evolutionary representations. A very interesting topic is a synthesis of CA by theoretical means, as contrasted to stochastic-search optimization. There are known methods for constructing finite-state automata as well as some theoretical work on 2D CA. This may also lead to modifying the transition function (the controller) – and using a different type of controller instead of MLP. We are also running more experiments to asses how much the representation suitability could scale up for more complex patterns.

Acknowledgment

Authors are thankful to Alexandre Devert for providing us with the source code for experiments from his dissertation, as well as for being always available to explain his approach. Part of this work was done with the support of Erasmus Exchange Programme.

References

[1] S. Wolfram, “A New Kind of Science”, Wolfram Media Inc., 2001. G.
[2] E. Berlekamp, J. Conway, R. K. Guy, “Winning Ways for Your Mathematical Plays”, Academic Press, 1982.
[3] M.Whitelaw, “Breeding Aesthetic Objects: Art and Artificial Evolution”, in Creative evolutionary systems, edited by P. J. Bentley and D. W. Corne, Academic Press, pp. 129 -145, 2002.
[4] H. de Garis, Artificial Embryology and Cellular Differentiation, in P. J. Bentley (ed.) Evolutionary design by computers, Morgan Kaufman, pp. 281-295, 1999.
[5] J. Romero, P. Muchado, The Art of Artificial Evolution, 2008.
[6] N. Hansen, The CMA Evolution Strategy: A Tutorial, 2008.
[7] J. F. Miller, Evolving a self-repairing, self-regulating, French flag organism, In GECCO, Springer Verlag, pp. 129—139, 2004.
[8] C. Kane and M. Schoenauer, Topological Optimum Design using Genetic Algorithms, Control and Cybernetics, 1996.
[9] G. S. Hornby, Generative representations for evolutionary design automation, Brandeis University, Waltham, MA, 2003, Ph.D. Dissertation.
[10] H. Hamda, F. Jouve, E. Lutton, M. Schoenauer and M. Sebag, Compact unstructured representations for evolutionary topological optimum design, Applied Intelligence, Volume 16, 2002
[11] A. Devert, Building processes optimization : Toward an artificial ontogeny based approach,Université Paris-Sud, France, 2009, Ph. D. Dissertation.
[12] J. Hlavačiková (Baran), “Cellular Embryogenic Representations in Evolutionary Design”, Master thesis, Comenius University, 2010.
[13] Universal Computer, LifeWiki, on-line at: http://conwaylife.com/wiki/index.php?title=Universal_computer accessed: July 12th 2010.
[14] K. Morita, M. Morgenstern, K. Imai, Universality of Reversible Hexagonal Cellular Automata, Theoret. Informatics Appl. 33, pp. 535-550, 1999.
[15] H. Hamda et al. Compact Unstructured Representations for Evolutionary Topological Optimum Design, Applied Intelligence 16, 139–155, 2002.