Proceedings of the 2007 IEEE Symposium on
Artificial Life (CI-ALife 2007)

Evolving an Artificial Regulatory Network for 2D Cell Patterning

Arturo Chavoya
University of Guadalajara
Periferico Norte 799
45000 Zapopan, Mexico
Email: achavoya@cucea.udg.mx

Yves Duthen
University of Toulouse 1
1, Place Anatole France
31042 Toulouse, France
Email: yves.duthen@univ-tlse1.fr

***Abstract*— Cell pattern formation has a central role in both artificial and natural development. This paper provides results from experiments in which a genetic algorithm (GA) was used to evolve an artificial regulatory network (ARN) to produce predefined bidimensional cell patterns through the selective activation of genes. The GA worked by evolving the gene regulatory network that was used to control cell reproduction. Cellular automata (CA) were chosen as models for cell patterning. After the final chromosomes were obtained, a single cell in the middle of the CA lattice was allowed to reproduce controlled by the ARN found by the GA, until the desired pattern was formed.**

I. Introduction

Computational Development is a relatively new sub-field of Evolutionary Computation that studies artificial models of cellular development, with the objective of understanding how complex structures and patterns can emerge from a small group of initial nonspecialized cells [1]. In biological systems, development is a fascinating and very complex process that involves following an extremely intricate program coded in the organism's genome.

One of the crucial stages of an organism's development is that of pattern formation, where the fundamental body plans of the individual are to be defined. It is now evident that gene regulatory networks play a central role in the development and normal functioning of living organisms [2]. It has been found that the different cell patterns created during the development of an organism are mainly due to the selective activation and inhibition of very specific regulatory genes.

Artificial Regulatory Networks (ARNs) are computer models that seek to imitate the gene regulatory networks found in nature. ARNs have previously been used to study differential gene expression either as a computational paradigm or to solve particular problems [3], [4], [5], [6], [7], [8]. On the other hand, evolutionary computation techniques have been extensively used in the past in a wide range of applications, and in particular they have previously been used to evolve ARNs to perform specific tasks [9], [10].

In this paper we describe research on using a genetic algorithm (GA) to evolve an ARN to create predefined bidimensional patterns by means of the selective activation of genes. The ARN used in this work is based on the model presented in [5]. In order to test the functionality of the ARN found by the GA, we applied the chromosomes representing the ARN to a cellular growth model that we have successfully used in the past to develop simple bidimensional and tridimensional geometrical shapes [11].

The paper starts with a section describing the cellular growth model, followed by a section presenting the ARN model and how it was implemented. The next section describes the GA used in this work and how it was applied to evolve the ARN. Results are presented next, followed by a section of conclusions.

II. Cellular Growth Model

Cellular automata (CA) were chosen as models of cellular growth, since they provide a simple mathematical model that can be used to study self-organizing features of complex systems [12]. CA are characterized by a regular lattice of N identical cells, an interaction neighborhood template η, a finite set of cell states Σ, and a space- and time-independent transition rule ϕ which is applied to every cell in the lattice at each time step.

In the cellular growth model presented in this work, a 33×33 regular lattice with non-periodic boundaries was used. The set of cell states was defined as $\Sigma = \{0, 1\}$, where 0 can be interpreted as an empty cell and 1 as an occupied or active cell. The interaction template η used was an outer Moore neighborhood. The CA's rule ϕ was defined as a lookup table that determined, for each local neighborhood, the state (empty or occupied) of the objective cell at the next time step. For a binary-state CA, these update states are termed the rule table's "output bits". The lookup table input was defined by the binary state value of cells in the local interaction neighborhood, where 0 meant an empty cell and 1 meant an occupied cell [13].

In this cellular growth model a cell can become active only if there is already an active cell in the interaction neighborhood. Thus, a new active cell can only be derived (reproduced) from a previously activated cell in the interaction neighborhood. Starting with an active cell in the middle of the lattice, the CA algorithm is applied allowing active cells to reproduce for 100 time steps according to the CA rule table. During an iteration of the CA algorithm, the order of reproduction of active cells is randomly selected in order to avoid artifacts caused by a deterministic order of cell reproduction.

1-4244-0701-X/07/$20.00 ©2007 IEEE

As mentioned above, the lattice was defined as a square of 33×33 cells, with the initial active cell at the central position. This particular lattice size was chosen in this and in previous work due to the minimum number of steps that the CA algorithm is required to run before an active cell reaches one of the lattice boundaries. This minimum number of steps was chosen originally as 15 because the GA chromosome coded in binary form the number of iterations that the CA algorithm was to run in order to generate a shape. This number was coded in 4 bits to give an upper limit of $2^4 - 1 = 15$ time steps [13]. For reasons of reproducibility, it was decided to use the same lattice size as in previous experiments. Furthermore, considering that cell reproduction normally follows an exponential trend, with a bigger lattice size the corresponding simulation times could significantly increase.

For all experiments, the CA were implemented as NetLogo models. NetLogo is a programmable modeling environment based on StarLogo that can be used to simulate natural and social phenomena [14]. It works by giving instructions to hundreds or thousands of independent "agents" all operating concurrently. It is well suited to study emergent properties in complex systems that result from the interaction of simple but often numerous entities. For each of the cell patterns studied, a NetLogo model was built.

III. Artificial Regulatory Network

An Artificial Regulatory Network (ARN) is a gene control model inspired by its biological counterpart. In nature, gene regulatory networks are widely used to control development and metabolic functions in living organisms [2]. Biological gene regulatory networks are a central component of an organism's genome, which is coded as one or more chains of DNA, and that interact with other macromolecules, such as RNA and proteins. Artificial genomes are usually coded as strings of discrete data types. The genome used in this work was implemented as a binary string starting with a series of regulatory genes, followed by a number of structural genes (see Fig. 1).

The gene regulatory network implemented is based on the ARN presented in [5]. However, unlike the ARN developed by this author, the genome implemented in the present work does not have promoter sequences and there are no unused intergene regions. All regulatory genes are adjacent and have predefined initial and end positions. Furthermore, the number of regulatory genes is fixed. On the other hand, the structural genes code for the particular shape grown by the reproducing cells and they were obtained using the methodology presented in [13]. Briefly, the cellular growth model described in Section II was evolved by a GA in order to produce predefined bidimensional shapes. The GA worked by evolving the CA rule table's output bits.

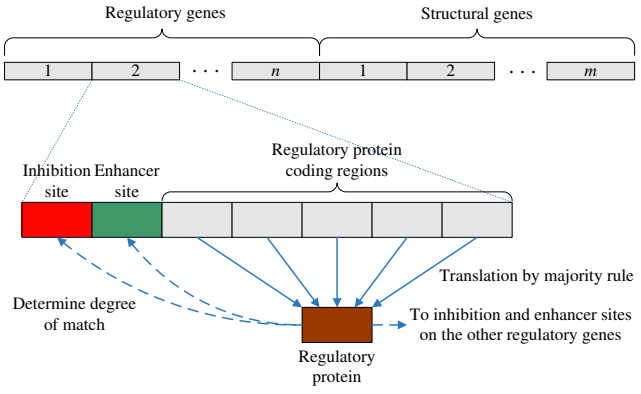

Fig. 1. Genome structure and regulatory gene detail.

As mentioned above, the ARN used in the present work consists of a series of regulatory genes, each of which consists of an inhibition site, an enhancer site and a series of regulatory protein coding regions (Fig. 1). The latter "translate" the coding regions into a regulatory protein using the majority rule, i.e. for each bit position in the protein coding regions, the number of 1's and 0's is counted and the bit that is in majority is translated into the regulatory protein. The inhibition site, the enhancer site and the individual protein coding regions all have the same size in bits. Thus the protein translated from the corresponding coding regions can be compared bit by bit with the inhibitor and enhancer sites and the degree of matching can be measured. As in [5], the comparison was implemented by an XOR operation, which results in a "1" if the corresponding bits are complementary.

Each translated protein is compared with the inhibition and enhancer sites of all the regulatory genes in order to determine the degree of interaction in the regulatory network. The influence of a protein on an enhancer or inhibitor site is exponential with the number of matching bits. The strength of excitation en or inhibition in for gene i with $i = 1, ..., n$ is defined as

$$en_i = \frac{1}{n} \sum_{j=1}^{n} c_j e^{\beta\left(u_{ij}^{+} - u_{\max}^{+}\right)} \tag{1}$$

$$in_i = \frac{1}{n} \sum_{j=1}^{n} c_j e^{\beta\left(u_{ij}^{-} - u_{\max}^{-}\right)}, \tag{2}$$

where n is the total number of regulatory genes, c_j is the concentration of protein j, β is a parameter set constant for all runs, u_{ij}^{+} and u_{ij}^{-} are the number of matches between protein j and the enhancer and inhibitor sites of gene i, respectively, and $u_{\max}^{+}$ and $u_{\max}^{-}$ are the maximum matches achievable between a protein and an enhancer or inhibition site, respectively [5].

Once the en and in values are obtained for all regulatory genes, the corresponding change in concentration for protein i in one time step is found using

$$\frac{dc_i}{dt} = \delta \left(en_i - in_i\right) c_i, \tag{3}$$

where δ is a constant that regulates the degree of protein concentration change.

Protein concentrations are updated and if a new protein concentration results in a negative value, the protein concentration

is set to zero. Protein concentrations are then normalized so that total protein concentration is always the unity.

As for structural genes, they are always associated to the corresponding regulatory genes. That is, structural gene number 1 is associated to regulatory gene number 1 and its corresponding translated protein, and so on. In the series of experiments presented in this work, the number of structural genes is always less than the number of regulatory genes. Thus, some regulatory proteins both regulate concentration for other proteins and directly control structural gene expression, while other proteins only have a regulatory role. Structural gene expression is visualized in the cellular growth model as a distinct external color for the cell.

A structural gene was defined as being active if and only if the regulatory protein translated by the associated regulatory gene was above a certain concentration threshold. The value chosen for the threshold was 0.5, since the sum of all protein concentrations is always 1.0, and there can only be a protein at a time with a concentration above 0.5. As a result, only one structural gene can be expressed at a particular time step in a cell. If a structural gene is active, then the CA lookup table coded in it is used to control cell reproduction.

Genome size in bits is dependent on the number and size of its component genes, and it was defined as

$$GenomeSize = n \times [(2 + k) \times r] + m \times s, \qquad (4)$$

where n is the number of regulatory genes, k is the number of regulatory protein coding regions, r is the region size in bits, m is the number of structural genes ($m \leq n$), and s is the structural gene size in bits. For all experiments we used the following parameter values: $n = 10$, $k = 5$, $r = 32$ and $s = 256$. The number of structural genes m took values from 1, 2, 3 or 6, depending on the experiment performed, as explained in section V. The number of regulatory genes n was chosen as 10 because after a series of experiments it was found that this value gave a desirable behavior for the protein concentrations. Parameter values for k and s are equal to those used in [5]. The value of 256 for parameter s results from the use of an outer Moore neighborhood for the CA lookup table that corresponds to the structural gene, that is $s = 2^8 = 256$ [13].

IV. Genetic Algorithm

Genetic algorithms are search and optimization methods based on ideas borrowed from natural genetics and evolution [15]. A GA starts with a population of chromosomes representing vectors in search space. Each chromosome is evaluated according to a fitness function and the best individuals are selected. A new generation of chromosomes is created by applying genetic operators on selected individuals from the previous generation. The process is repeated until the desired number of generations is reached or until the desired individual is found.

The GA in this paper uses tournament selection as described in [16] with single-point crossover and mutation as genetic operators. Single-point crossover consists in randomly selecting two chromosomes with a certain probability called crossover rate, and then randomly selecting a single bit position in the chromosome structure. From this point on, the remaining fragments of the two chromosomes are exchanged. The resulting chromosomes then replace the original ones in the chromosome population. On the other hand, mutation consists in randomly flipping one bit in a chromosome from 0 to 1 or vice versa. The probability of each bit to be flipped is called the mutation rate.

After several calibration experiments, we settled for the following parameter values. The initial population consisted of 1000 binary chromosomes chosen at random. Tournaments were run with sets of 3 individuals randomly selected from the population. Crossover and mutation rates were 0.60 and 0.15, respectively. Finally, the number of generations was set at 50, since there was no significant improvement after this number of generations. The crossover rate of 0.60 was chosen equal to that used in previous work [11], [13], because it was reported to give the best results when trying to evolve a binary string representing a CA using a GA [17]. As for the mutation rate, we decided to use a value one order of magnitude higher than the one used in the same report, for the reasons that will be given in Section V.

The fitness function used by the GA was defined as

$$Fitness = \frac{1}{k} \sum_{i=1}^{k} \frac{ins_i - \frac{1}{2} outs_i}{des_i}, \qquad (5)$$

where k is the number of different colored shapes, each corresponding to an expressed structural gene, ins_i is the number of filled cells inside the desired shape i with the correct color, $outs_i$ is the number of filled cells outside the desired shape i, but with the correct color, and des_i is the total number of cells inside the desired shape i. Thus, a fitness value of 1 represents a perfect match. This fitness function is an extension of the one used in [18], where the shape produced by only one "gene" was considered. To account for the expression of several structural genes, the combined fitness values of all structural gene products were introduced in the fitness function used.

In this work, a GA chromosome corresponds to the ARN portion of the binary genome. Chromosome size was thus dependent on the parameter values used. The ARN used in this work has a size of 2240 bits divided in 10 regulatory genes, which represents a search space of $2^{2240} \approx 2 \times 10^{674}$ vectors. It should be evident that this search space is far too large for any sort of exhaustive evaluation. Search space grows exponentially with the number of regulatory genes. But even for the simplest of ARNs, that consisting of only two regulatory genes, the search space has a size of $2^{448} \approx 7 \times 10^{134}$, which is still too large to be explored deterministically.

During the course of a GA experiment, each chromosome produced in a generation was fed to the corresponding NetLogo model, where the structural genes were attached and the cells were allowed to reproduce controlled by the ARN found by the GA. Fitness was evaluated after the model

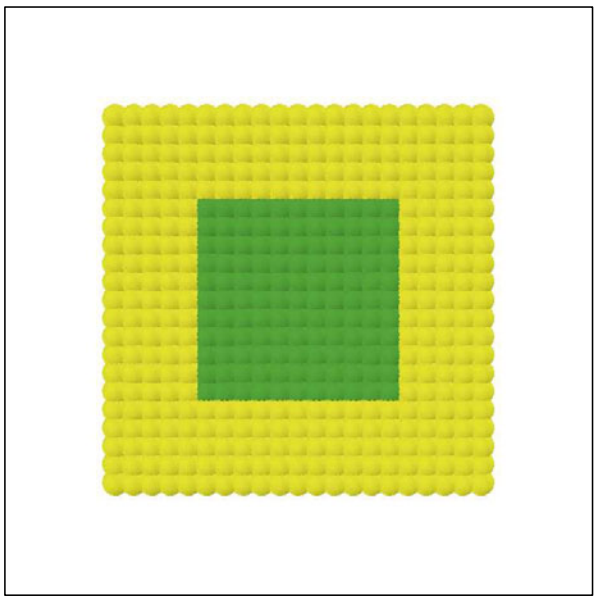

Fig. 2. Two-color square.

stopped and a colored shape was formed. This process continued until the maximum number of generations was reached and the best individual was obtained.

V. Results

In all cases, the GA described in Section IV was used to evolve the ARN for the desired colored shapes. After an ARN was obtained, an initial active cell in the middle of the CA lattice was allowed to reproduce (sprout an active cell from a previously active cell) controlled by the gene activation sequence found by the GA. In the desired shapes studied, each color represents a different structural gene being expressed. In order to achieve the desired shape with a predefined color for each cell, the genes in the ARN had to evolve to be activated in a precise sequence and for a specific number of iterations.

As a first step, in order to find the appropriate values for parameters β and δ used in the ARN, a series of GA experiments was performed. The tests consisted in finding an ARN to grow a 21×21 square in the cellular growth model in at most 10 generations, using 10 regulatory genes and one structural gene that coded for the square. The other parameters values for the GA and the ARN were set as described in the preceding sections.

Being a factor to an exponent, parameter β could not be varied widely. Thus, a range of 0.5 to 5.0 in increments of 0.5 units was tried. For parameter δ, a range of 1.0×10^0 to 1.0×10^{20} with exponent increments of one unit was used. Surprisingly, parameter δ was found to be very flexible in the range of values that could successfully find the desired shape. For a β value of 1.0, runs with parameter δ in the range from 1.0×10^6 to 1.0×10^{20} found the correct shape under the conditions described. At the end, the values for β and δ were set to 1.0 and 1.0×10^6, respectively.

Once the appropriate parameter values were found, several series of experiments were performed in order to evolve ARNs for various colored shapes. Not all GA experiments rendered an ARN capable of forming the desired shape. Furthermore, for some desired shapes involving the expression of more than three structural genes, no appropriate ARN could be evolved. The graphs presented next correspond to some of those experiments where ARNs with fitness function values equal to 1.0 were found by the GA.

Fig. 2 shows a two-color 21×21 square built by the expression of two structural genes, both coding for a square. The graph corresponding to the expression of the regulatory proteins of the evolved ARN is presented in Fig. 3. For some of the graphs shown, only the first 60 time steps are presented, as there is no significant change in these graphs after this point and until the end of the 100-step run.

When trying to create a three-color square expressing three identical structural genes each coding for a square, it proved difficult to synchronize the expression of the three regulatory genes in a specific sequence. In order to increase the likelihood for the GA to find an appropriate ARN, instead of using a series of three structural genes coding for a square, a tandem of two series of three structural genes was used, for a total of six structural genes. In that manner, for creating the inner square, the ARN could express either structural gene number 1 or gene number 4, for the middle square it could use genes 2 or 5, and finally for the outer square it could make use of structural genes 3 or 6. Thus, the probability of finding an ARN that could express a three-color square with a particular color order was substantially increased. Using an ARN with this configuration of structural genes, the three-color 25×25 square shown in Fig. 4 was obtained. The graph representing the expression of the proteins for the corresponding ARN is shown in Fig. 5. Note the order in which the structural genes were expressed, starting with the first gene from the second series of structural genes, followed by the second gene from the first series, and ending with the third gene from the second series of structural genes.

These shapes were chosen so that the different structural

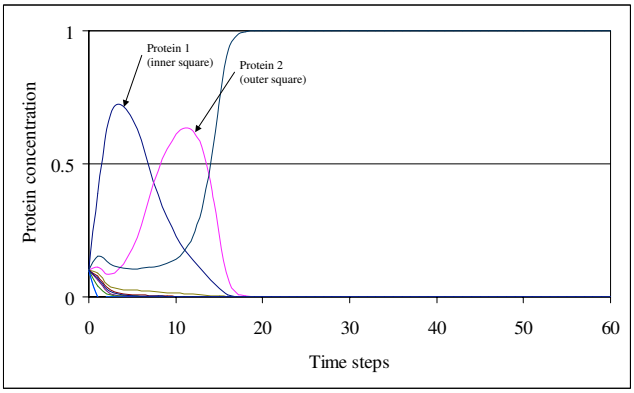

Fig. 3. Graph for the protein concentrations from an ARN expressing the two-color square.

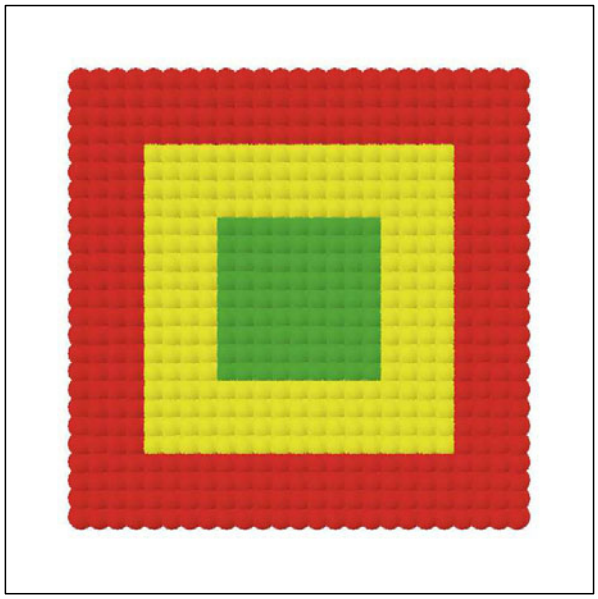

Fig. 4. Three-color square.

genes were expressed for the same number of time steps in the cellular growth model. For the two-color square, each structural gene is expressed for five time steps, whereas for the three-color square, each of the three genes involved is activated for exactly four time steps.

In order to explore the result of combining different structural genes that are expressed for a different number of time steps, three different genes were used to grow a French flag. One gene drove the creation of the central white square, while the other two genes extended the central square to the left and to the right, expressing the blue and the red color, respectively. The last two structural genes do not code specifically for a square, instead they extend a vertical line of cells to the left or to the right for as many time steps as they are activated. The problem of generating a French flag pattern was first introduced by Wolpert in the late 1960s [19] while trying

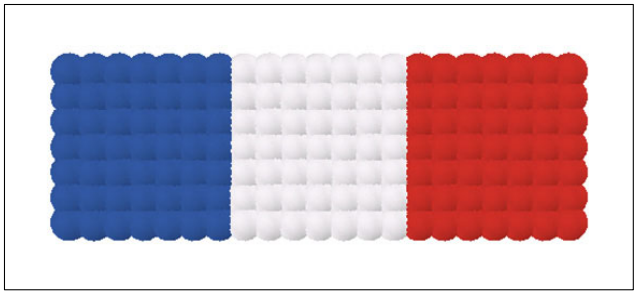

Fig. 6. French flag (21×7).

to formulate the problem of cell pattern development and regulation in living organisms, and it has been used since then by some authors to illustrate the problem of artificial pattern development [20].

Unlike the two- and three-color squares, where each gene had to be activated in a precise sequence, to create the flag the central square could be extended to the left or to the right in any of the two orders, that is, first extend to the left and then to the right, or vice versa. This allowed more flexibility for the GA to find an appropriate ARN. Figure 6 shows a 21×7 French flag grown from the expression of the three structural genes mentioned above. The graph of the corresponding ARN protein concentration pattern is shown in Fig. 7. After the white central square is formed, the left blue square is grown, followed by the right red square.

To illustrate a different sequence of gene activation, the 27×9 French flag shown in Fig. 8 was created. The corresponding protein concentration graph is presented in Fig. 9. Note that in this case, the right red square is formed before the left blue one.

When the shapes generated do not require strictly sequential gene activation, structural genes do not necessarily have to complete one shape before forming another shape. Take as example the case of the French flag shape, where some structural genes can interrupt their expression and then resume activation at a later time after the expression of another

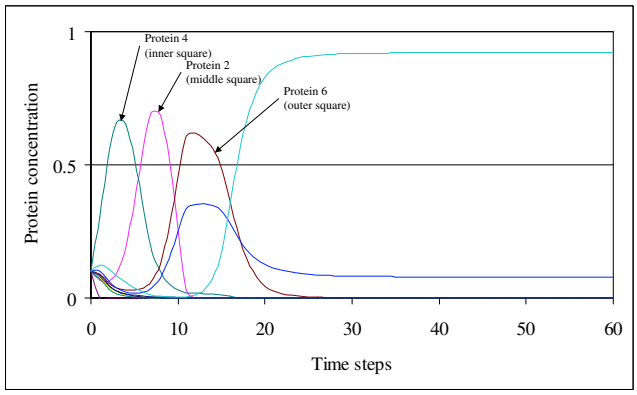

Fig. 5. Graph for the protein concentrations from an ARN expressing the three-color square.

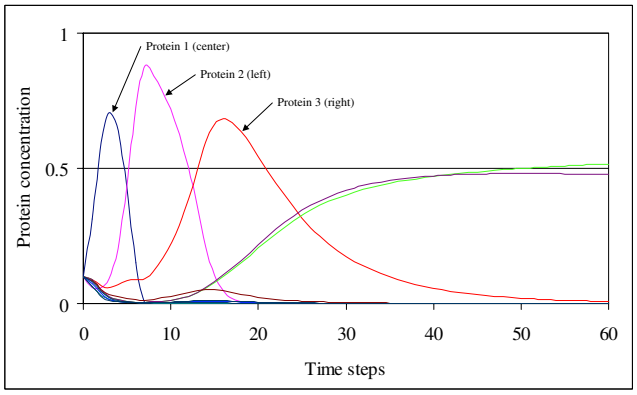

Fig. 7. Graph for the protein concentrations from an ARN expressing the 21×7 French flag.

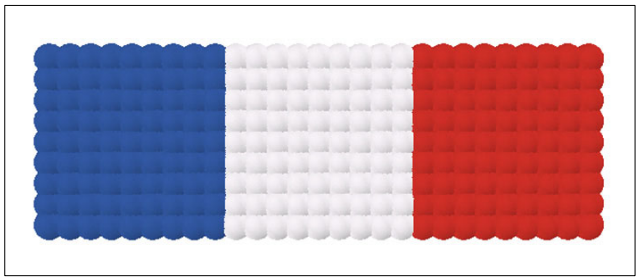

Fig. 8. French flag (27 × 9).

gene. One of the genes that extend cells to one side can be interrupted to allow the gene that extends cells to the opposite side to be activated, and then resume its activation to finish the shape. Fig. 10 shows the protein concentration graph of one of these cases, where a 27×9 French flag is generated. After the white central square is formed, the gene that extends the red cells to the right is activated for only 5 time steps generating a 5×9 red rectangle. Then the gene that extends the blue cells to the left is activated for 9 time steps until completion of the left blue square. Finally, the gene that extends cells to the right becomes activated again for another 4 time steps to finish the generation of the red right square.

One point worth noting regarding the GA parameters is that a relatively high mutation rate was used. This choice was made since it was found that single bits could have a considerable influence in the final behavior of the ARN. As an example, consider the two graphs shown in Fig. 11 from an experiment where a two-color square was grown. The corresponding ARNs for these graphs differ only in the value of bit position 952 on the enhancer site of regulatory gene number 5. While the upper graph of Fig. 11 corresponds to a chromosome with fitness value of 0.50, the lower graph corresponds to a chromosome with a fitness value of 0.93. In fact, further mutation of this latter chromosome on bit 599 in regulatory gene number 3, gets the ARN to achieve a fitness value of 1.00.

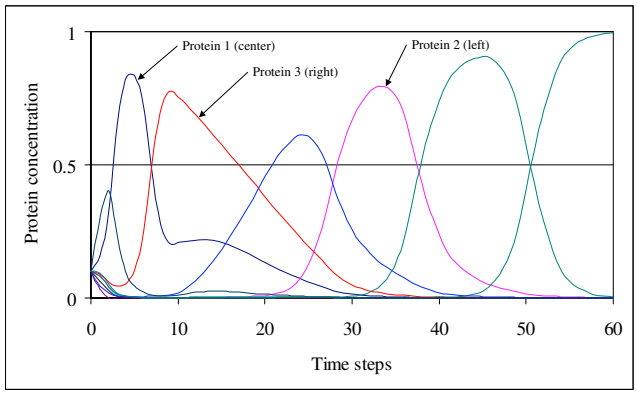

Fig. 9. Graph for the protein concentrations from an ARN expressing the 27×9 French flag.

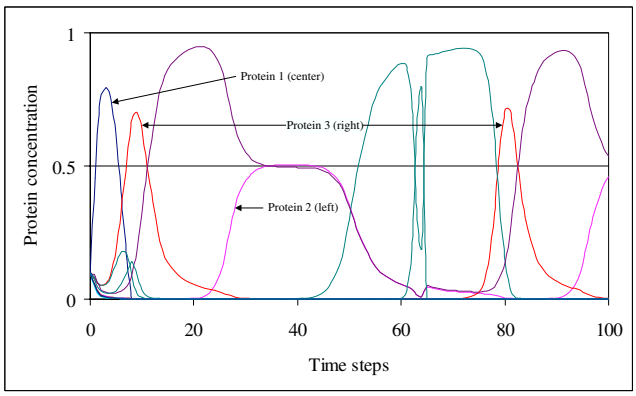

Fig. 10. Graph for the protein concentrations from an ARN expressing the 27×9 French flag built from the alternating activation of structural genes.

VI. Conclusion

The experiments presented in this paper show that a GA can give reproducible results in evolving an ARN to grow predefined bidimensional cell patterns starting with a single cell. In particular, it was found that using this ARN model it is feasible to synchronize reliably up to three genes, although some problems were found when trying to synchronize the activation of more than three genes in a precise sequence.

In general, the framework used proved to be suitable for obtaining simple patterns involving the activation of two or three genes, but more work is needed to explore pattern formation of more complex forms, both in 2D and 3D. It is also desirable to search for an ARN model that can reliably synchronize the activation of more than three genes. Furthermore, in order to

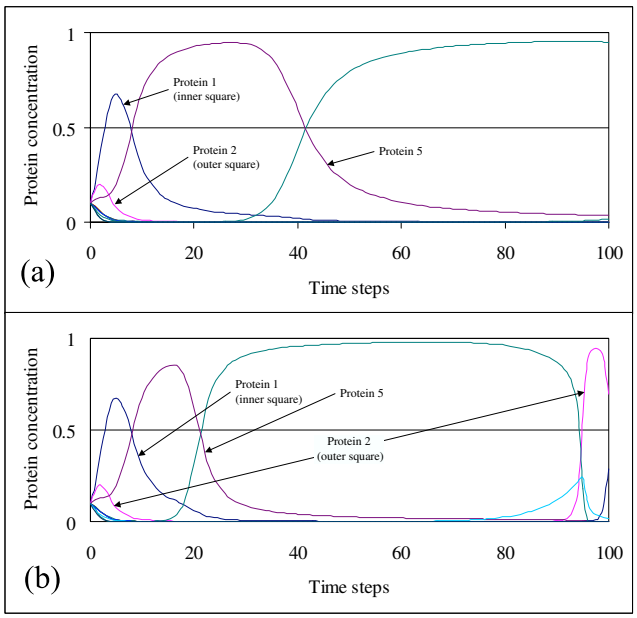

Fig. 11. Effect of a single different bit in the ARN. (a) Fitness 0.50; (b) Fitness 0.93

build a more accurate model of the cell patterning process, interaction with the environment and other artificial entities may be necessary.

To the authors' knowledge, this is the first report of the use of a GA to evolve a specific behavior in an ARN to grow an artificial cell pattern. Previous attempts at evolving ARNs have been made using other evolutionary computation techniques. A simple GA was chosen in this work for evolving the ARN due to the discrete and fixed-size nature of the artificial genome used. Moreover, it was considered that the GA was the evolutionary computation paradigm that resembled the most the actual evolutionary mechanism seen in nature.

The long-term goal of this work is to study the emergent properties of the artificial development process. It can be envisioned that one day it will be feasible to build highly complex structures arising mainly from the interaction of myriads of simpler entities controlled by a development program.

REFERENCES

[1] S. Kumar and P. J. Bentley, "An introduction to computational development," in *On Growth, Form and Computers*, S. Kumar and P. J. Bentley, Eds. New York, NY, USA: Academic Press, 2003, ch. 1, pp. 1–44.

[2] E. H. Davidson, *The Regulatory Genome: Gene Regulatory Networks in Development And Evolution*, 1st ed. Academic Press, 2006.

[3] P. Eggenberger, "Evolving morphologies of simulated 3D organisms based on differential gene expression," in *Proceedings of the 4th European Conference on Artificial Life*, I. Harvey and P. Husbands, Eds. Springer, 1997, pp. 205–213.

[4] T. Reil, "Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny," in *Proceedings of the 5th European Conference on Artificial Life (ECAL)*. New York, NY: Springer Verlag, 1999, pp. 457–466.

[5] W. Banzhaf, "Artificial regulatory networks and genetic programming," in *Genetic Programming Theory and Practice*, R. L. Riolo and B. Worzel, Eds. Kluwer, 2003, ch. 4, pp. 43–62.

[6] P. D. Kuo and W. Banzhaf, "Small world and scale-free network topologies in an artificial regulatory network model," in *Proceedings of Artificial Life IX, (ALIFE-9), Boston, USA*, J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. Watson, Eds. MIT Press, Cambridge, 2004, pp. 404–409.

[7] F. Stewart, T. Taylor, and G. Konidaris, "Metamorph: Experimenting with genetic regulatory networks for artificial development." in *ECAL*, 2005, pp. 108–117.

[8] N. Flann, J. Hu, M. Bansal, V. Patel, and G. Podgorski, "Biological development of cell patterns: Characterizing the space of cell chemistry genetic regulatory networks." in *ECAL*, 2005, pp. 57–66.

[9] J. Bongard, "Evolving modular genetic regulatory networks," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*. IEEE Press, Piscataway, NJ, 2002, pp. 1872–1877.

[10] P. D. Kuo, A. Leier, and W. Banzhaf, "Evolving dynamics in an artificial regulatory network model," in *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN-04)*, ser. LNCS 3242, X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervs, J. Bullinaria, J. Rowe, P. Tino, A. Kabán, and H.-P. Schwefel, Eds. Birmingham, UK: Springer, Berlin, September 2004, pp. 571–580.

[11] 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*. New York, NY, USA: ACM Press, 2006, pp. 231–232.

[12] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of Modern Physics*, vol. 55, pp. 601–644, 1983.

[13] A. Chavoya and Y. Duthen, "Evolving cellular automata for 2D form generation." in *Proceedings of the Ninth International Conference on Computer Graphics and Artificial Intelligence 3IA'2006*, 2006, pp. 129–137.

[14] U. Wilensky, "NetLogo," http://ccl.northwestern.edu/netlogo/, 1999, center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[16] M. Mitchell, *An introduction to genetic algorithms*. Cambridge, MA, USA: MIT Press, 1996.

[17] R. Breukelaar and T. Bäck, "Using a genetic algorithm to evolve behavior in multi dimensional cellular automata: emergence of behavior." in *GECCO '05*, 2005, pp. 107–114.

[18] H. de Garis, "Artificial embryology and cellular differentiation," in *Evolutionary Design by Computers*, P. J. Bentley, Ed. San Francisco, USA: Morgan Kaufmann Publishers, Inc., 1999, pp. 281–295.

[19] L. Wolpert, "The French flag problem: a contribution to the discussion on pattern development and regulation," in *Towards a Theoretical Biology*, C. Waddington, Ed. New York, NY, USA: Edinburgh University Press, 1968, pp. 125–133.

[20] J. F. Miller and W. Banzhaf, "Evolving the program for a cell: from French flags to Boolean circuits," in *On Growth, Form and Computers*, S. Kumar and P. J. Bentley, Eds. Academic Press, Oct. 2003.