Evolving a Self-Repairing, Self-Regulating, French Flag Organism

1 author:

Julian Francis Miller
The University of York
316 PUBLICATIONS   **7,766** CITATIONS

Some of the authors of this publication are also working on these related projects:

Project   Upscaling the "reservoir computing in-materio" hardware View project

Project   Discovering human-readable algorithms with Cartesian Genetic Programming View project

Evolving a self-repairing, self-regulating, French flag organism

Julian Francis Miller

Department of Electronics, University of York, Heslington, York, UK, YO10 5DD
jfm@ohm.york.ac.uk
http://www.elec.york.ac.uk/intsys/users/jfm7/

Abstract. A method for evolving programs that construct multicellular structures (organisms) is described. The paper concentrates on the difficult problem of evolving a cell program that constructs a fixed size French flag. We obtain and analyze an organism that shows a remarkable ability to repair itself when subjected to severe damage. Its behaviour resembles the regenerative power of some living organisms.

1 Introduction

The development of a fully formed adult from the zygote has to rank as one of the most remarkable feats of molecular engineering in the universe. From a single set of instructions inside one cell, an organism can grow to contain 10^{13} cells (for a human being) containing many hundreds of specialized cells performing distinct functions. How does nature achieve this feat of engineering? In his book biologist Frank M. Harold explains [9]:

> "Genes specify the cell's building blocks; they supply raw materials, help regulate their availability and grant the cell independence of its environment. But the higher levels of order, form and function are not spelled out in the genome. They arise by the collective self-organization of genetically determined elements, affected by cellular mechanisms that remain poorly understood."

The genotype-phenotype mapping employed by nature is highly complex and many-to-one. Despite this, in many branches of evolutionary algorithms genetic representations make no distinction between genotype and phenotype. This is a drawback if one is interested in problems that involve phenotypes of arbitrary size and complexity [1]. If higher level organisms were really colonies of cells with different genotypes it would have been much harder for evolution to evolve organisms of the complexity and sophistication of many living creatures. The poor scalability of directly encoded systems (i.e. a one-to-one mapping from genotype to phenotype) is particularly evident in the evolution of neural

networks, where each link requires a floating-point weight that must be determined. The work presented in this paper builds on the author's previous work in evolving multicellular organisms [13]. This paper is devoted to a particularly difficult but interesting problem, that of the growth and regulation of a differentiated multicellular organism that looks like a French flag. We examine in particular two evolved solutions that achieve growth regulation. One of the solutions appears to be static but shows a remarkable ability for self-repair that is reminiscent of the regenerative ability of some living organisms. The other solution shows interesting internal time dynamics. It is not the aim of this work to model closely natural developmental processes, but rather, to explore a simple idealization of biological development in the hope that it will exhibit some of the advantages of biological systems. The long term aim of this work is to investigate a new way of constructing software and hardware systems that are self-repairing and can achieve levels of complex and intelligent behaviour that top-down design methods are unable to attain.

The plan for the paper is as follows: A review of related work is given in section 2. Section 3 describes how the cells and their environment are represented, and the cell program's inputs and outputs. Section 4 describes the form of genetic programming used to evolve the cell program. Section 5 describes the experiments and the results obtained. In section 6, a solution is analyzed in detail. Section 7 examines the organism's behaviour under various kinds of damage. The paper concludes and discusses future work.

2 Related Work

Fleischer and Barr created a sophisticated multicellular developmental test bed and included realistic models of chemical diffusion, cell collision, adhesion and recognition [6]. Their purpose was to investigate cell pattern generation. They noted that the design of an artificial genotype that develops into a specific pattern is very difficult. They also noted that size regulation is critical and non-trivial and that developmental models tend to be robust to perturbations. Eggenberger suggests that the complex genotype-phenotype mappings typically employed in developmental models allow the reduction of genetic information without losing the complex behaviour. He stresses the importance of the fact that the genotype will not necessarily grow as the number of cells, thus he feels that developmental approaches will scale better on complex problems [5]. Bongard and Pfeifer have evolved genotypes that encode a gene expression method to develop the morphology and neural control of multi-articulated simulated agents [3]. Bentley and Kumar examined a number of genotype-phenotype mappings on a problem of creating a tessellating tile pattern [2]. They found that the indirect developmental mapping (that they refer to as an implicit embryogeny) could evolve the tiling patterns much quicker, and further, that they could be subsequently grown to (iterated) much larger sized patterns. One drawback that they reported was that the implicit embryogeny tended to produce the same types of pat-

terns. Other researchers are more motivated by fundamental biological aspects of cell behaviour. Furusawa and Kaneko modeled cell internal dynamics and its relationship to the emergence of cell multicellularity[7]. Hogeweg has carried out impressive work in computer models of development and constructed a sophisticated model of cells (biotic) by modeling the internal dynamics by groups of cells in a cellular automaton that are subject to energy minimization [10][11]. The energy minimization leads to cell movement and sorting by differential cell adhesion. The cell genome was modeled as 24 node Boolean network that defined cell signaling and adhesion. She used a fitness criterion that was related to the difference in the gene expression in all the cells. She evolved organisms that exhibited many behaviours that are observed in living systems: cell migration and engulfing, budding and elongation, and cell death and re-differentiation. Streichert et al. have investigated the problems of growth regulation and self-repair in artificial embryos with a single cell type using Random Boolean Networks and S-Systems [15]. Recently, many of the research contributions in computational development have been presented in a single volume [12].

3 Cell and chemical representation

The cell's genotype is a representation of a feed-forward Boolean circuit (that implements the cell program). This maps the cell's input conditions to output behaviour. A cell is a square in a non-toroidal two-dimensional cellular automaton. The inputs to each live cell program are bits defining the cells states and the chemicals in the Moore neighbourhood. Using this information, the cell's program decides on the amounts of each chemical that it will produce (as binary bits), whether it will live, die, or change to a different cell type at the next time step, and how it will grow over the Moore neighbourhood. It also decides a single bit that represents whether it will obey the grow bits or not. This output was introduced to make it easier for a cell program to decide not to grow.
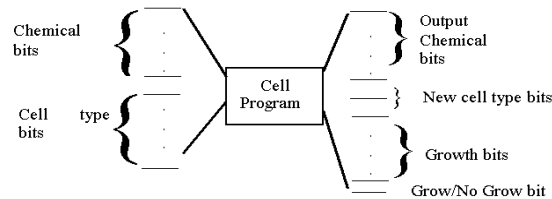

Fig. 1. The cell program's binary inputs and outputs

Unlike real biology, when a cell replicates itself, it is allowed to grow in any or all of the eight neighbouring cells simultaneously (this is done to speed up growth, mainly for reasons of efficiency). In all the experiments reported in this paper the amount of each

chemical is represented by an eight-bit binary number. The cell types are represented by binary codes with zero reserved for the absence of a cell (or dead) and are synonymous with cell colour (1 - blue, 2-red, 3-white). In general, the user can decide how many cell types there are (a power of two), in the experiments reported here, only four cell types were required and also how many chemicals there are. Only live cells have their programs executed. Initially a single cell is placed in the grid (the zygote). If two or more cells decide to grow into the same location at the next time step, the last such cell in the scan path overwrites all previous growths. This was chosen as it greatly simplified the process of constructing the newly grown organism, though of course, it introduces a bias that isn't present in a truly parallel system. The process of constructing the new organism at time t+1 from the organism at time t is the following: Every live cell from the top-left to the bottom-right has its program run (all cells run the same program). A new map (initially empty) is created and filled with cells that have either grown, or not died, in the map at time t. After all the programs inside the living cells have been run, the map at time t+1 replaces the map at time t. For each chemical there is a rectangular array of the same dimensions and type as the cellular map. Chemicals obey a diffusion rule defined as follows: let N denote the neighbourhood with neighbouring position k,l, the chemical at position i,j at the new time step is given by (1).

$$(c_{ij})_{new} = 1/2(c_{ij})_{old} + \frac{1}{16}\sum_{k,l \in N}(c_{kl})_{old}. \tag{1}$$

This ensures that over time, chemicals diffuse away from their point of origin. The rule was designed so that diffusing chemical would be conserved (apart from the loss when the level falls below a level of one). Note that, since cells can determine their own new level of chemical there is no strict conservation of chemical in the entire system (i.e. a cell with its chemicals overwrite those at the location that they grow into). The chemical map is scanned and updated in a similar manner to the cellular map. A depiction of the cell's inputs and outputs is shown in Fig. 1.

4 Cartesian Genetic Programming and the Cell Program

Cartesian Genetic Programming was developed from methods developed for the automatic evolution of digital circuits [14]. CGP represents a program or circuit as a list of integers that encode the connections and functions. The representation is readily understood from a small example. Consider the one bit binary adder circuit (Fig. 2). This has three inputs that represent the two bits to be summed and the carry-in bit. It has two outputs: sum and carry-out. CGP employs an indexed list of functions that represent in this example, various two input logic gates and three input multiplexers. Suppose that in a function lookup table AND is function 6, XOR is function 10 and MUX is function 16.

The three inputs A, B, Cin are labeled 0, 1, 2. The output of the left (right) XOR gate is labeled 3 (6). The output of the MUX gate is labeled 5. The AND output is labeled 4. In Fig. 2, a genotype is shown and how it is decoded to a phenotype (the one-bit binary adder). The integers in italics represent the functions, and the others represent the connections between gates, however, if it happens to be a two input gate then the third input is ignored. It is assumed that the circuit outputs are taken from the last two nodes. The second group of four integers (shown in grey) represent an AND gate (with output 4) that is not part of the circuit phenotype. Since only feed-forward circuits are being considered, it is important to note that the connections to any gate can only refer to gates that appear on its left.
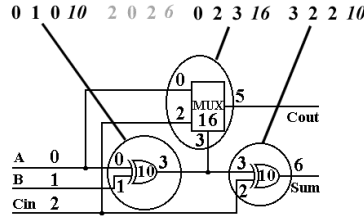
Fig. 2. The Cartesian genotype and corresponding phenotype for a one-bit adder circuit

Typically, CGP uses point mutation (that is constrained to respect the feed-forward nature of the circuit). Suppose that the first input of the MUX gate (0) was changed to 4. This would connect the AND gate into the circuit (defined by the four grey genes). Similarly, a point mutation might disconnect gates. Thus, CGP uses a many to one genotype-phenotype mapping, as redundant nodes may be changed in any way and the genotypes would still be decoded to the same phenotype. The (1+4)-ES evolutionary algorithm uses characteristics of this genotype-phenotype mapping to great advantage (i.e. genetic drift):

1. Generate 5 chromosomes randomly to form the population
2. Evaluate the fitness of all the chromosomes in the population
3. Determine the best chromosome (called it *current_best*)
4. Generate 4 more chromosomes (offspring) by mutating the *current_best*
5. The *current_best* and the four offspring become the new population
6. Unless stopping criterion reached return to 2

Step 3 is a crucial step in this algorithm: if more than one chromosome is equally good then the algorithm always chooses the chromosome that is not the *current_best* (i.e. equally fit but genetically different). This step allows a genetic drift process that turns out be of great benefit [16][18]. The mutation rate is defined to be the percentage of each chromosome that is mutated in step 4. In all the experiments described in this paper only four kinds of MUX logic gates were employed defined by the expression f(A,B,C)=AND(A, NOT(C)) OR AND(B, C). The four types correspond to cases where

inputs A and B are either inverted or not. Program outputs are taken from consecutive nodes at the end of the phenotype with the leftmost of these being the grow/no grow output.

5 Evolutionary experiments and results

In the biological development of organisms, cells have to behave differently according to their position within the organism. Lewis Wolpert [17] proposed that this positional information arises from a combination of intercellular interactions and cellular responses to chemical gradients that form relative to organism boundaries. In the model, cells respond differently according to threshold concentrations of chemicals. He likened the problem to one of growing a French Flag; the developmental method of construction would be able to produce a recognizable flag of arbitrary size. This illustrates an important property of developmental systems in that they are scale free (i.e. there is no relationship between the genotype size and the size of the phenotype). Wolpert's model was the inspiration for the task the maps of cells were to achieve. In this paper we describe experimental results for evolving organisms whose task is to grow from a single zygote into the 63 cell organism that looks like a French flag by a certain time (i.e. achieve maturity). This is a very difficult task as initially the cell program must replicate to grow to the desired size but must also somehow recognize that it has reached the appropriate size and no longer continue to replicate. It must also output the desired cell state signal (represented by colour) in the correct spatial region of the flag. The cell program (as with real embryos) is not given coordinates but must decide how to act by local interactions only. This is shown in Fig 3.
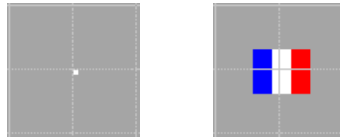

Fig. 3. Task definition: a single cell program beginning from a white state, at time 0, must replicate itself and differentiate itself into other cells, so that at iteration 6, it becomes a 63 cell French flag, and remains like that indefinitely. Initially, at the same location as the single start cell, there can be chemicals having values 0 or 255

The evolved organism and target organism were compared at iterations 6, 7, 8, 9, cell by cell, and a cumulative score of correctness (fitness) was calculated. By presenting exactly the same target at these four test points, we hoped to steer the evolution of the cell program towards growing into a fixed size French flag organism. The cell's Cartesian program was allowed 300 nodes and 20 runs of 30,000 generations were carried out (with 1% mutation) for five chemical scenarios: no chemicals up to four allowed chemicals. The amount of chemical initially present at the location of the zygote was initialized as fol-

lows: the first chemical is given 255, the second given 0, the third given 255, and the fourth given 0. The maximum fitness value is 1024, which occurs when all four 16 by 16 cell maps for the organism match perfectly with the target organism. A table of results for the five chemical scenarios is shown in Table 1. All solutions with fitness above 975 were iterated over 20 iterations to ascertain whether any of them stopped growing. Only two solutions were found with this property. The first was the solution with the highest overall fitness with two defined chemicals. The temporal behavior of this is shown in Fig. 4.
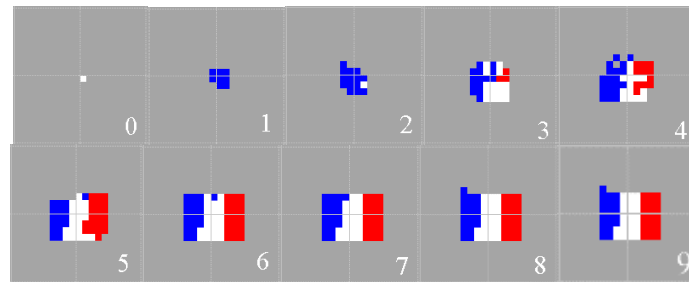
Fig. 4. Growth of fittest cell program from a white seed cell to a mature French flag (two chemicals)

The other solution that stopped growing occurred in the three chemical scenario, however, surprisingly it had fitness 988 (fourth best). This is shown in Fig. 5. Unlike the best overall solution this shows complex time dependent behaviour indefinitely (it was iterated over 50 iterations and did not grow). The results shown in Table 1. show that having chemicals makes it easier to find fitter organisms, and the results indicate that having either none or one chemical make it unlikely, if not impossible, to achieve solutions that grow and then stop growing that meet the target objective.

Table 1. Performance statistics for French flag problem. The maximum fitness is 1024

#chemicals	Average final best fitness (20 runs)	Standard deviation	Best fitness	Worst fitness	Average number of cell program nodes	#nodes in fittest program
0	875.85	16.41	909	855	155.05	153
1	924.80	30.48	975	862	153.90	165
2	938.65	31.00	1012	894	145.75	139
3	948.80	27.87	1008	907	146.45	144
4	941.50	20.35	988	918	142.95	139

The fourth fittest solution moves through a repeating cycle of activity indefinitely (as far as can be determined) always remaining bounded within a small region and with the majority of cells remaining the same.
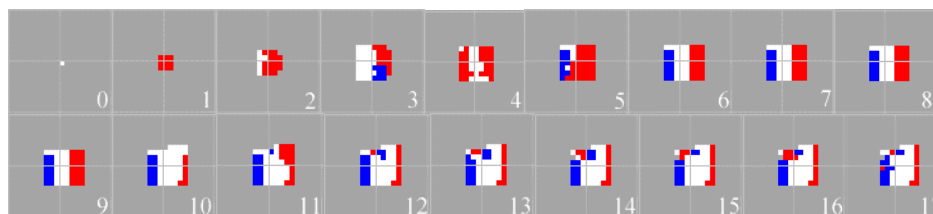
Fig. 5. Growth of fourth fittest cell program from a white seed cell (three chemicals)

6 Analysis

It is instructive to examine how the fittest French flag organism achieves apparent stasis at iteration 8 and subsequently. The cell program itself is too large and complicated to be shown in this paper: from Table 1 we see that it uses 139 binary IF statements (multiplexers). In addition the program cannot be understood without referring to the current state of the organism. However, in the compass of this paper we can illuminate aspects of the developmental program by showing (Fig. 6) the decisions being made by each cell in the organism. On the left, we see the organism itself at iteration 8 and on the right the cell growth and replication fate map. To clarify its interpretation, consider the line of four grey cells in the fate map that are live cells in the organism. These cells will obey the growth instructions at the next iteration (see Table 2) but will then die immediately after-

ward. Clearly as the organism remains unchanged, the cells around them grow over them, thus reconstructing the line of cells as if nothing had happened.
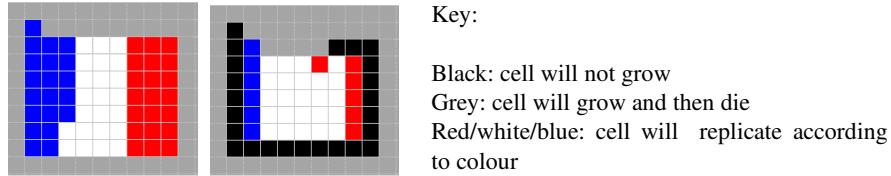

Key:

Black: cell will not grow
Grey: cell will grow and then die
Red/white/blue: cell will replicate according to colour

Fig. 6. Cell growth/replication fate map for the French flag organism at iteration 8.

Table 2. Cell growth instructions for actively growing cells in fittest French flag at iteration 8.

N	E	S	W	NE	SE	SW NW
01101111	01001101	01001101	01001101	11011100		
01101111	01101100	01101100	01101100	01010100	01010101	11010101
01101111	01101100	01101100	01101100	01110100	01110101	11010101
01101111	01101100	01101100	01101100	01110100	01110101	11010101
01101101	01101100	01101100	01101100	01110100	01110101	11010101
01101101	01101100	01101100	01101100	01110100	01110101	11010101

To clarify the interpretation of the cell growth instructions in Table 2, consider the bottom right actively growing red cell, this has growth instructions 11010101. According to Fig. 6, it replicates its own colour in the directions: N, E, W, SE and NW. All the interior cells of the flag (within one cell inside) are actively growing and overwriting each other (according the top-left, bottom-right scan path), while most of the border cells will remain as they are. Note that none of the second column of blue cells in the flag overwrite their blue left neighbours, as they do not replicate to the west. The French flag is clearly being actively reconstructed even though it appears to be static.

7 Autonomous behaviour after damage

When the maturing French flag is damaged, it is often able to regenerate itself (or produce another similar French flag) in some cases, though sometimes it can be put into a state of continuous growth (further examples in Table 3). Fig. 7 shows the regeneration of a French flag cellular map from the original white central region of the original (fittest cell program). The cells replicate and then start to differentiate; eventually the growth of the organism slows and stops at iteration 20.
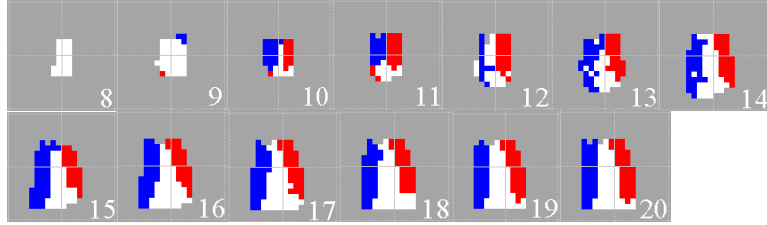
Fig. 7. Autonomous recovery of badly damaged French flag organism conditions (blue and red regions killed at iteration 8 - see Fig. 4). There is no further change after iteration 20

Fig. 8 shows what happens when the original cells of the fittest French flag are placed in a random but contiguous arrangement - chemical maps left intact - the phenotype grows a little at first and rapidly re-organizes over a period of time, eventually reaching stasis (by iteration 24). This behaviour is reminiscent of autonomous regeneration of the pond organism hydra, which can reform itself when its cells are dissociated and then re-aggregated in a centrifuge [8].
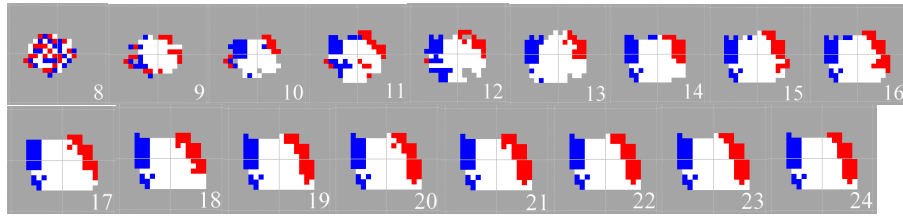
Fig. 8. Autonomous recovery of French flag from randomly rearranged cells (French flag at iteration 8 - see Fig. 4). There is no further change after iteration 24

In Table 3, we show the behaviour of the fittest French flag organism after it is disrupted or damaged at iteration 8. In many cases it achieves stability again fairly rapidly and recovers the approximate appearance of the French flag, however when the damage is too severe (as with a large hole or 25% cells disrupted randomly) the organism undergoes continuous growth and doesn't appear to stabilize (even when run for many more iterations). Such dynamic processes are hard to control in all circumstances and it illustrates the enormously difficult balancing act that living systems have to carry out.
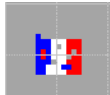
8 Conclusions and further work

We have presented and investigated an idealized model of development and studied in detail the growth and regulation of an organism made of cells that can replicate, differen-

tiate, and read and produce chemicals through local interactions. There are many avenues for further investigation such as, the roles of overwriting in self-repair and the diffusion law of chemicals, complexity of evolvable structures, and the evolvability of the representation. The software written also allows the possibility of cell movement (prior to growth) but as yet, this hasn't been investigated. There are many ways that the system can be made more sophisticated, for instance, by allowing cells to control the flow of chemicals and to add the possibility of cell adhesion. However, since the eventual aim of the work is toward technological applications it is important to try to keep the model as simple as possible. The great robustness of the evolved organisms to damage may be a consequence of the ability of cells to overwrite each other; this remains for further investigation.

Work is already underway in examining the possibility of using the developing organisms for robot control, thus giving the organism a function. It will be interesting to see if control programs can recover autonomously after damage. A detailed investigation also needs to be undertaken about the chemical information that is provided to the cell's pr o-gram. It was discovered when the software was written to carry out this work, that the author inadvertently only provided the most significant bit of each chemical to the cell's program (i.e. cell's think chemicals are either high or low) and all the other bits were read as zeros. It has been found that providing all chemical bits or providing only the most significant bit made it very difficult, if not impossible, to solve the tasks presented. This is such an interesting finding that it warrants lengthy and detailed future investigation. This will be reported in due course.

Table 3. Behaviour of fittest French flag when damaged at iteration 8.

| Initial condition | Final condition (iterations to stability below) |

References

1. Banzhaf, W., Miller J. F.: The Challenge of Complexity. In: Menon, A. (ed.): Frontiers of Evolutionary Computation. Kluwer Academic Publishers (2004)
2. Bentley, P. J., Kumar S.: Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem. In: Proceedings of the Congress on Evolutionary Computation, IEEE Press (1999) 35-43
3. Bongard, J. C., Pfeifer R.: Repeated Structure and Dissociation of Genotypic and Phenotypic Complexity in Artificial Ontogeny. In: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan-Kaufmann, (2001) 829-836
4. Dellaert, F.: Toward a Biologically Defensible Model of Development, Masters thesis, Dept. of Computer Eng. and Science, Case Western Reserve University (1995)
5. Eggenberger, P.: Evolving morphologies of simulated 3D organisms based on differential gene expression, In: Proceedings of 4th European Conf. on Artificial Life (1997) 205-213
6. Fleischer, K., Barr, A. H.: A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In Langton C. G (ed.) Proceedings of the 3rd Workshop on Artificial Life, Addison-Wesley (1992) 389-416
7. Furusawa, C., Kaneko, K.: Emergence of Multicellular Organisms with Dynamic Differentiation and Spatial Pattern. In: Adami C. et al. (eds.) Proceedings of the 6th International Conference on Artificial Life, MIT Press (1998)
8. Gierer, A., Berking, S., Bode, H., David, C. N., Flick, K., Hansmann, G., Schaller, H., Trenkner E.: Regeneration of hydra from reaggregated cells, Nature New Biology, Vol. 239 (1972) 98-101
9. Harold, F. M.: The Way of The Cell. Oxford University Press (2001)
10. Hogeweg, P.: Evolving Mechanisms of Morphogenesis: on the Interplay between Differential Adhesion and Cell Differentiation, J. Theor. Biol., Vol. 203 (2000) 317-333
11. Hogeweg, P.: Shapes in the Shadow: Evolutionary Dynamics of Morphogenesis, Artificial Life, Vol. 6 (2000) 85-101
12. Kumar, S., Bentley P. J. (eds.): On Growth, Form and Computers, Academic Press (2003)
13. Miller, J. F.: Evolving Developmental Programs for Adaptation, Morphogenesis and Self-Repair. In: Proceedings of the 7th European Conf. on Advances in Artificial Life. LNAI, Vol. 2801 (2003) 256-265
14. Miller, J. F., Thomson, P.: Cartesian Genetic Programming. In: Proceedings of the 3rd European Conf. on Genetic Programming. LNCS, Vol. 1802 (2000) 121-132
15. Streichert, F., Spieth, C., Ulmer, H., Zell, A.: Evolving the Ability of Limited Growth and Self-Repair for Artificial Embryos. In: Proceedings of the 7th European Conf. on Advances in Artificial Life. LNAI, Vol. 2801 (2003) 289-298
16. Vassilev, V. K., Miller J. F.: The Advantages of Landscape Neutrality in Digital Circuit Evolution. In: Proceedings of 3rd Int. Conf. on Evolvable Systems: From Biology to Hardware, LNCS, Vol. 1801, Springer-Verlag (2000) 252-263
17. Wolpert, L.: Principles of Development. Oxford University Press (1998)
18. Yu, T., Miller, J. F.: Neutrality and the evolvability of Boolean function landscape. In: Proceedings of the 4th European Conference on Genetic Programming, Springer-Verlag (2001) 204-217