Evolving Developmental Programs for Adaptation, Morphogenesis, and Self-Repair

Julian F. Miller

School of Computer Science, University of Birmingham, Birmingham, UK, B15 2TT
j.miller@cs.bham.ac.uk
http://www.cs.bham.ac.uk/~jfm

Abstract. A method for evolving a developmental program inside a cell to create multicellular organisms of arbitrary size and characteristics is described. The cell genotype is evolved so that the organism will organize itself into well defined patterns of differentiated cell types (e.g. the French Flag). In addition the cell genotypes are evolved to respond appropriately to environmental signals that cause metamorphosis of the whole organism. A number of experiments are described that show that the organisms exhibit emergent properties of self-repair and adaptation.

1 Introduction

The mechanisms whereby mature multicellular biological organisms are constructed from the original fertilized egg are extraordinary and complex. Living organisms are perpetually changing yet outwardly they often give the appearance of stasis. This is accomplished by cells constantly reproducing and dying. There is no central control behind this. Such a process leads naturally to the self-repairing properties of organisms. At present humans design systems using a top down design process. Until now, this methodology of design has served humanity well. However when one looks at the problem of constructing software and hardware systems with of the order of 10^{13} or more components (roughly the number of cells in the human body) it appears that we face a design crisis. As the number of interacting components grows the complexity of the system grows also. This can lead to many problems. It becomes difficult to formally verify such systems due to the combinatorial explosion of configurations or states. The cost of maintenance grows alarmingly also and their unreliability increases. Electronic circuits are now routinely constructed at sub-micron level. This is leading to immense wiring problems. As the trend for increasing miniaturization of circuits continues, it will become ever more difficult to supply the huge amounts of data required to define or program these devices. Living systems demonstrate very clever solutions to these problems. The information defining the organism is contained within each, and every, part. There is no central design. Consequently, it is likely that designers will have to increasingly turn to methods of constructing systems that mimic the developmental process that occur in living systems. To some extent, such an agenda has been embraced in the nascent field of amorphous computing [1].

Genetic programming is a paradigm in which computer program are subjected to process that mimics Darwinian evolution [19]. The dominant representation of programs chosen is that of a tree in which there is no distinction between genotype and phenotype. This is a drawback if one is interested in problems that involve phenotypes of arbitrary size and complexity. Natural evolution works on the level of the genotype that is formed when an ovum is fertilized to form a zygote. This undergoes an extraordinary process of cell replication and differentiation to construct an entire organism. 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. Thus, it seems imperative that the conventional GP paradigm must be extended to encompass development. 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 is motivated by an number of questions: (1) How can we define programs that run inside cells that construct complex structures of arbitrary size, when each cell runs an identical program? (2) Is it possible to evolve organisms that can adapt to environmental signals? (3) Can organisms be evolved that are capable of self-repair? 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. It is hoped that computer science might benefit from such studies.

The plan for the paper is as follows: A review of relevant 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 some conclusions and ideas for future work are given.

2 Related Work

Fleischer and Barr created a sophisticated multi-cellular developmental test bed and included realistic models of chemical diffusion, cell collision, adhesion and recognition [9]. 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.

A number of researchers have studied the potential of Lindenmeyer systems [20] for developing artificial neural networks (ANNs) and generative design. Boers and Kuiper have adapted L-systems to develop the architecture of artificial neural networks (ANNs) (numbers of neurons and their connections) [4]. They used an evolutionary algorithm to evolve the rules of a L-system that generated feed-forward neural networks. Backpropagation was used, and the accuracy of the neural networks on test data was assigned to the fitness of the encoded rules. They found that this method produced more modular neural networks that performed better than networks with a predefined structure. Kitano developed another method for evolving the architecture of an artificial neural network [17] using a matrix re-writing system that manipulated adjacency matrices. Although Kitano claimed that his method produced superior results to direct methods (i.e. a fixed architecture, directly encoded and evolved), it was later shown in a more careful study that the two approaches were of equal quality [23]. Gruau devised an elegant graph re-writing method called cellular encoding [11]. Cellular encoding is a language for local graph transformations that controls the division of cells that grow into artificial neural networks. This was shown to be an effective method for optimizing both the architecture and weights at the same time, and they found that, to achieve as good performance with direct encoding, required the testing of many candidate architectures [12]. Others have successfully employed this approach in the evolution of recurrent neural networks that control the behaviour of simulated insects [18]. Recently Hornby and Pollack have also evolved context free L-systems to define three dimensional objects (table designs) [15]. They found that their generative system could produce designs with higher fitness and faster, than direct methods. Jacobi created an impressive artificial genomic regulatory network, where genes code for proteins and proteins activate (or suppress) genes [16]. He used the proteins to define neurons with excitatory or inhibitory dendrites. This allowed him to define a recurrent ANN that was used to control a simulated Khepera robot for obstacle avoidance and corridor following. Nolfi and Parisi evolved encoded neuron position and branching properties of axonal trees that would spread out from the neurons and connect to other neurons [22] and in later work introduced cell division using a grammar [6]. Astor and Adami have created a developmental model of the evolution of an ANN that utilizes an artificial chemistry [2]. 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 [8]. Sims evolved the morphology and behaviour of simulated agents [24]. Bongard and Pfeifer have evolved genotypes that encode a gene expression method to develop the morphology and neural control of multi-articulated simulated agents [5]. Bentley and Kumar examined a number of genotype-phenotype mappings on a problem of creating a tessellating tile pattern [3]. 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 patterns. Other researchers are more motivated by fundamental biological aspects of cell behaviour. Furusawa and Kaneko are modeled cell internal dynamics and its relationship to the emergence of cell multi-cellularity[10]. Hogeweg has carried out impressive work in computer models of development and constructed a sophisticated model of cells (biotic) by modeling their internal dynamics by groups of cells in a cellular automaton that are subject to energy minimization [13][14]. 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.

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. Each live cell sees its own state and the states of its eight immediate neighbours. It also sees the amount of chemical (at present a single chemical) at the location of each of its eight neighbours. Using this information, the cell's program decides on the amount of chemical that it will produce, whether it will live, die, or change to a different cell type at the next time step, and how it will grow.

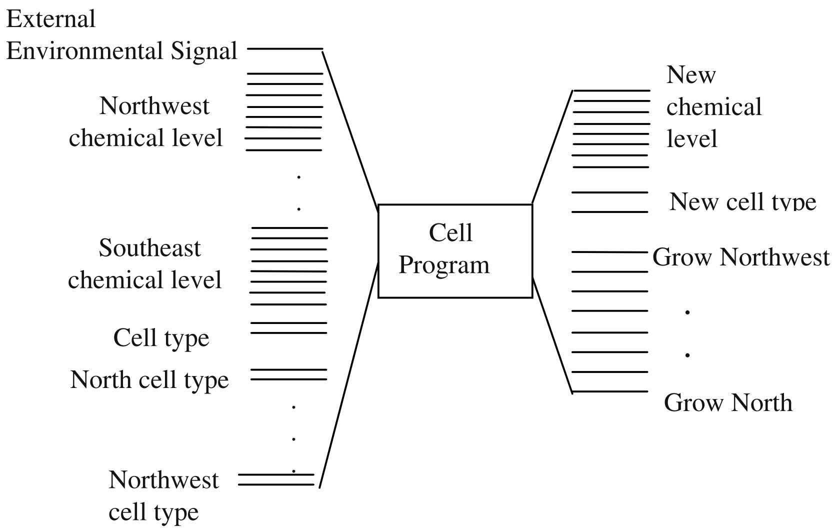

Fig. 1. If an external environmental signal is (not) defined a cell's program reads 83 (82) binary bits: 64 bits represent the amount of chemical at the eight immediate neighbours and 18 bits represent all the cell types in the Moore neighbourhood. There are 18 output bits in the program: eight for the amount of chemical that will be placed at the locations where the replicated cell exists at the next time step, two for the new cell type, and eight bits that define the places where the new cell can grow

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 there are three cell types (blue, red, white) and the amount of chemical is represented by an eight-bit binary number. The cell types are represented by two-bit binary codes: 00 = dead or absence of a cell, 01 = blue cell, 10 = red cell, 11 = white cell. Only live cells have their programs executed. Initially a single cell is placed in the grid (the zygote). In the experiments reported in this paper, the zygote has been chosen to be white In addition an amount of chemical equal to 255 (the maximum) is placed at the zygote's location. 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. The two dimensional grid is scanned from the top-left corner to the bottom right. 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. The chemical map is updated in a similar manner. A depiction of the cell's inputs and outputs is shown in Fig. 1. The chemicals obey the diffusion rule (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 cell's can determine their own new level of chemical there is no strict conservation of chemical in the entire system. 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).

4 Cartesian Genetic Programming and the Cell Program

Cartesian Genetic Programming was developed from methods developed for the automatic evolution of digital circuits [21]. 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 (5). The output of the MUX gate is labeled 6. 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 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.

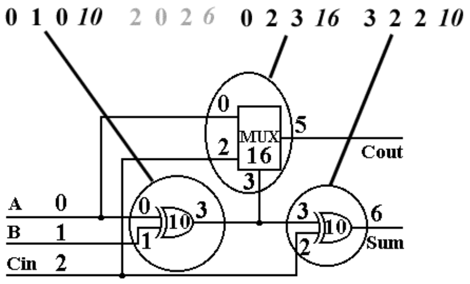

Fig. 2. One-bit adder circuit.

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). This is given below:

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 [25][27]. 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. Since in this paper 18 output bits are defined, the rightmost 18 multiplexers are used. The first of these defines the most significant bit of the new chemical level.

5 Evolutionary Experiments and Results

The French Flag model of Lewis Wolpert [26] was the inspiration for the task the maps of cells were to achieve. The following experiments were carried out.

1. Produce a growing and recognizable French flag
2. Produce a growing pattern of blue spots
3. Produce an adaptive growing pattern of blue or red spots

In the first experiment two organisms representing French flags were defined. These were used to compare the evolved organism at particular times in the development of the organism. The evolved organism and target organism were compared cell by cell and a cumulative score of correctness was calculated. This was the fitness of the cell genotype used in the evolutionary algorithm.

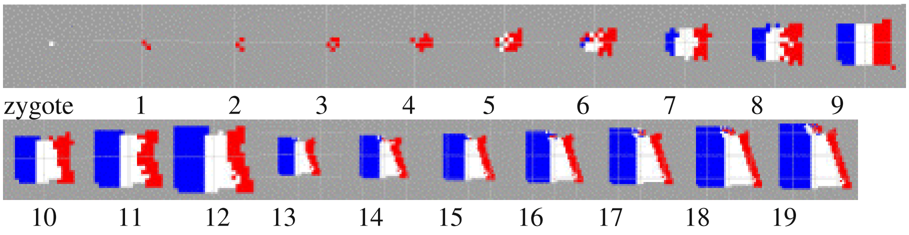

Fig. 3. Best solution for task 1 (fitness test points at time = 7 and 9)

When the organism produced in task 1 is iterated beyond iteration 9 it begins to deform a little (Fig. 3.). However it still remains recognizable as the French flag. In a further post-evolution experiment the organism in task 1 was subjected to severe damage. Firstly, a large rectangular section of the organism at iteration 9 was removed and the organism allowed to develop. Secondly, the white and red sections were removed. Fig. 4 shows how the organism is able to reconstruct itself, though it retains some scarring. Note the images changes scale at iteration 13 (Figs. 3 and 4).

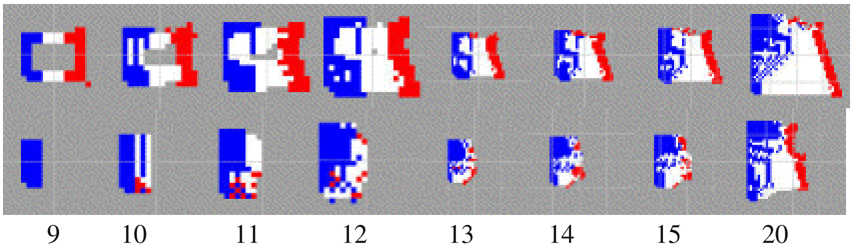

Fig. 4. The organism for task one is damaged but repairs itself

The second experiment required the evolved organism to produce a growing (and always faithful) pattern of blue rectangular spots surrounded by white cells.

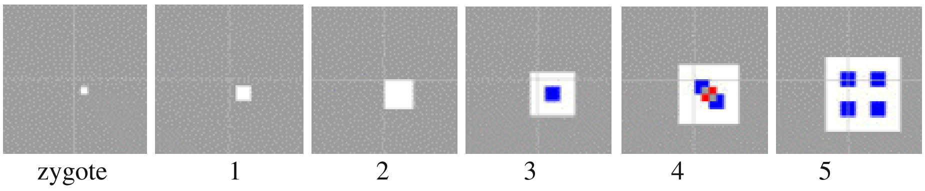

Fig. 5. A perfect solution for task 2 (single fitness test point at time = 5). The temporal pattern alternates indefinitely between blue spots and an elegant tiling pattern.

The third experiment utilized the external environmental signal and required the evolved organism to be a growing blue spots pattern when the signal was zero and a growing red spots pattern when the signal was one. The fitness of the evolved organism was compared against the desired blue and red spots pattern and a fitness computed in the usual manner. The experimental parameters were: population = 5, generations = 30,000, runs = 10, mutation rate = 1%, max number of multiplexers = 200.

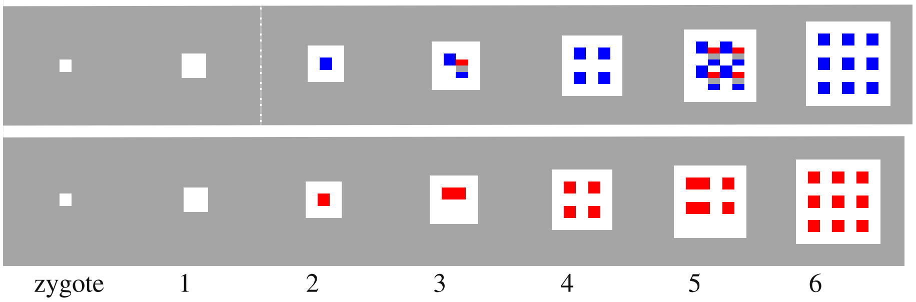

Fig. 6. A perfect solutions for task 3 (fitness test points at time = 4 and 6)

The behaviour of the adaptive organism in task 3 changes when exposed to new environmental signal that it hadn't encountered during evolution. Fig. 7 shows that for an alternating environmental signal (0 then 1 alternately) the organism adopts a stable but very different behaviour. If the environmental signal takes other temporal sequences of 0 and 1 much more chaotic behaviour is obtained. If the chemical is changed, or the chemical program bits ignored, the organisms behaviour can be radically affected. In the case of task 3 the chemical map behaved in a very different way depending on the environmental signal. Organisms evolved without chemicals were of much lower fitness and lacked stability. The 18 programs (see Fig. 1) that produce each of these behaviours have been examined and are difficult to interpret. Given the 83 binary input bits it would be fiendishly difficult for a human designer to create programs that meet the defined requirements. Detailed analysis of these will be published in due course.

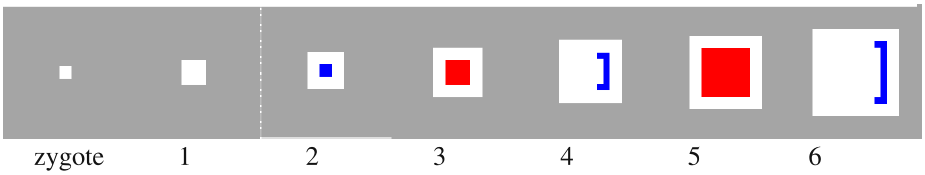

Fig. 7. Behaviour of task 3 organism with alternating environmental signal

6 Conclusions and Further Work

A developmental genotype has been described that can under evolution, and solve tasks with phenotypes of arbitrary size capable of self-repair and adaptation under global environmental change. In the future, coordinated cell growth and movement will be investigated. Careful work remains to be undertaken to examine the role of chemicals. Future investigations will consider the chemical as energy so that cell growth and death might become an emergent phenomenon, this might also be helpful in establishing a more open ended evolution, where multi-cellular organisms fight for survival, thus linking morphology with behaviour.

References

1. Abelson H., Allen D., Coore D., Hanson C., Homsy G., Knight Jr. T., Nagpal R., Rauch E., Sussman G. J., Weiss R. (1999), "Amorphous Computing", MIT Tech. report, AI Memo 1665.
2. Astor J. C., and Adami C. (2000), "A Development Model for the Evolution of Artificial Neural Networks", Artificial Life, Vol. 6, pp. 189–218.
3. Bentley P., and Kumar S. (1999), "Three ways to grow designs: A comparison of embryogenies for an Evolutionary Design Problem", in Proceedings of the Congress on Evolutionary Computation, IEEE Press, pp. 35–43.
4. Boers, E. J. W., and Kuiper, H. (1992), "Biological metaphors and the design of modular neural networks", Masters thesis, Department of Computer Science and Department of Experimental and Theoretical Psychology, Leiden University.
5. Bongard J. C. and Pfeifer R. (2001), "Repeated Structure and Dissociation of Genotypic and Phenotypic Complexity in Artificial Ontogeny", in Spector L. et al. (eds.) Proceedings of the Genetic and Evol. Comput. Conference, Morgan-Kaufmann, pp. 829–836.
6. Cangelosi, A., Parisi, D., and Nolfi, S. (1993), "Cell Division and Migration in a 'Genotype' for Neural Networks", Tech. report PCIA-93, Inst. of Psych., CNR, Rome.
7. Dellaert, F. (1995), "Toward a Biologically Defensible Model of Development", Masters thesis, Dept. of Computer Eng. and Science, Case Western Reserve University.
8. Eggenberger P. (1997), "Evolving morphologies of simulated 3D organisms based on differential gene expression", Proc. Of European Conf. on Artificial Life, pp. 205–213.
9. Fleischer, K., and Barr, A. H. (1992), "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, pp. 389–416.
10. Furusawa C., and Kaneko, K. (1998), "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.
11. Gruau, F. (1994), "Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm", PhD thesis, Ecole Normale Supérieure de Lyon.
12. Gruau, F., Whitley, D., and Pyeatt, L. (1996) "A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks", in Proc. of the 1st Annual Conference on Genetic Programming, Stanford.
13. Hogeweg, P. (2000), "Evolving Mechanisms of Morphogenesis: on the Interplay between Differential Adhesion and Cell Differentiation", J. Theor. Biol., Vol. 203, pp. 317–333.

14. Hogeweg, P. (2000), "Shapes in the Shadow: Evolutionary Dynamics of Morphogenesis", Artificial Life, Vol. 6, pp. 85–101.
15. Hornby G. S., and Pollack J. B. (2001), "The Advantages of Generative Grammatical Encodings for Physical Design", in Proceedings of the Congress on Evolutionary Computation, IEEE Press, pp. 600–607.
16. Jacobi, N. (1995), "Harnessing Morphogenesis", Cognitive Science Research Paper 423, COGS, University of Sussex.
17. Kitano, H. (1990), "Designing neural networks using genetic algorithms with graph generation system", Complex Systems, Vol. 4, pp. 461–476.
18. Kodjabachian, J. and Meyer, J-A. (1998), "Evolution and Development of Neural Controllers for Locomotion, Gradient-Following and Obstacle-Avoidance in Artificial Insects", IEEE Transactions on Neural Networks, Vol. 9, pp. 796–812.
19. Koza, J. R.: Genetic Programming: On the programming of computers by means of natural selection. MIT Press (1992), Genetic Programming II: Automatic Discovery of Reusable Subprograms. MIT Press (1994)
20. Lindenmeyer, A. (1968), "Mathematical models for cellular interaction in development, parts I and II", Journal of Theoretical Biology, Vol. 18, pp. 280–315.
21. Miller, J. F. and Thomson, P. (2000), "Cartesian genetic programming", in Proceedings of the 3rd European Conf. on Genetic Programming. LNCS, Vol. 1802, pp.121–132.
22. Nolfi, S., and Parisi, D. (1991) "Growing neural networks", Technical report PCIA-91-15, Institute of Psychology, CNR, Rome.
23. Siddiqi, A. A., and Lucas S. M. (1998), "A comparison of matrix rewriting versus direct encoding for evolving neural networks", in Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE Press, pp. 392–397.
24. Sims K. (1994), "Evolving 3D morphology and behaviour by competition", in Proceedings of Artificial Life IV, pp. 28–39.
25. Vassilev V. K., and Miller J. F. (2000), "The Advantages of Landscape Neutrality in Digital Circuit Evolution", 3rd Int. Conf. on Evolvable Systems: From Biology to Hardware, LNCS, Vol. 1801, Springer-Verlag, pp. 252–263.
26. Wolpert, L. (1998), "Principles of Development", Oxford University Press.
27. Yu, T. and Miller, J. (2001), "Neutrality and the evolvability of Boolean function landscape", in Proceedings of the 4th European Conference on Genetic Programming, Springer-Verlag, pp. 204–217.