Adaptive Fractal Gene Regulatory Networks for Robot Control

Peter J. Bentley

Department of Computer Science, University College London, Gower Street, London.
P.Bentley@cs.ucl.ac.uk

Abstract. Fractal proteins are an evolvable method of mapping genotype to phenotype through a developmental process, where genes are expressed into proteins comprised of subsets of the Mandelbrot Set. The resulting network of gene and protein interactions can be designed by evolution to produce specific patterns that in turn can be used to solve problems. In this paper, adaptive developmental programs, capable of developing different solutions in response to different signals from an environment, are investigated. Experiments show that such methods are highly effective in producing robot controllers that generate different movements in response to sensor inputs.

1 Introduction

Development in nature never ceases. From conception to the death of the organism, genes are expressed into proteins, which control cellular growth, differentiation and pattern formation. This remarkable process produces highly complex structures from minimal numbers of genes. For example, a current estimate of the number of genes in the human genome is 20,000 – and a current estimate of the number of neurons in our brains is 100 billion.

But development doesn't just generate complex phenotypes. It also enables those phenotypes to regenerate (repair themselves if damaged) and adapt. Plants provide the most obvious examples of adaptive development. Even if two plants are genetically identical, they will grow differently, dynamically adapting to their environments. The differences between phenotypes can be extreme, as a cursory examination of any ivy tree will show.

In a very real sense, all organisms rely on adaptive development. Each cell is controlled by the proteins produced by the genes inside it, but those genes are affected by signals (proteins produced by genes) from other cells. Change the signals, and the development of cells is changed, dynamically and adaptively. (Wolpert et al, 2001)

In animals those signals come mainly from within (although forces such as gravity are essential for correct skeletal and musculature development, light is essential for the correct development of the visual cortex, and temperature can affect the gender of many reptiles). In plants, the external environment also provides factors such as sunlight, forces (e.g., an obstacle blocking the direction of growth) and nutrients in the soil. These factors are often translated into proteins which trigger major changes in development, for example "shade leaves" instead of "sun leaves". (Wolpert et al 2001)

Whether we regard the external environment of an organism as the source of "extended" cell signals (cf. Dawkins's extended phenotype), or we regard intercellular signals as an environment for the cell, it is clear that the ability for development to respond adaptively (and differently) to different environmental signals is critical.

The work described here continues an ongoing investigation into the use of fractals as a computer representation of proteins. Earlier work has shown that fractal proteins are highly evolvable by a genetic algorithm (Bentley 2004, 2003c), that specific patterns of activation in a fractal gene regulatory network (GRN) can be evolved (Bentley, 2004, 2003b), that evolved fractal GRNs naturally show fault-tolerance (Bentley 2003c), and that they can perform computational tasks such as function regression and robot control (Bentley 2003a). This work now focuses on adaptive GRNs, posing the following question: Can a single fractal developmental process be evolved, which generates different results in different environments? The experiments in this paper provide evidence to show that such adaptive GRNs can be evolved quickly, reliably and consistently, by producing wall-following robot controllers that make a robot respond differently, given different sensor inputs.

2 Background

Researchers such as Hornby (2003), Bongard (2002), and Kumar and Bentley (2003) have demonstrated that various types of development can enable smaller genotypes to represent more complex phenotypes through the ability of development to discover modularities and repetition. Other scientists in the field have been focussing on the ability of developmental methods to enable self-repairing behaviour and graceful degradation of solutions. For example, the work of Andy Tyrrell and his group create fault-tolerant hardware inspired by ideas of embryology and immune systems (Jackson and Tyrrell, 2002). More recently, Julian Miller has described experiments evolving developmental programs to create "French Flag" patterns (Miller and Banzhaf, 2003). He shows that development is able to regenerate these patterns, and that different patterns can be evolved in different environments. Work on applying developmental algorithms to robot control is less common. Most seem to rely on the development of neural networks that are then used to control motion (and in some cases generate form) (Jakobi 1995; Hornby, 2003; Bongard 2002). However, the recent work of Quick (2003) is highly relevant, describing the evolution of a simple GRN for control of a Khepera robot. Quick emphasises the need for GRNs to be embodied (linked or coupled) to their environments, with changes in environment causing changes in the GRN (Quick 2003). He successfully evolved light-finding behaviour, although there seem to be evolvability issues with his simple representation of genes and proteins.

3 Fractal Proteins

Development is the set of processes that lead from egg to embryo to adult. Instead of using a gene for a parameter value as we do in standard EC (i.e., a gene for long legs), natural development uses genes to define proteins. If expressed, every gene generates a specific protein. This protein might activate or suppress other genes, might be used for signalling amongst other cells, or might modify the function of the cell it lies within. The result is an emergent "computer program" made from dynamically forming gene regulatory networks (GRNs) that control all cell growth, position and behaviour in a developing creature (Wolpert et al, 2001).

Table 1. Types of objects in the representation

fractal proteins	defined as subsets of the Mandelbrot set.
Environment	contains one or more fractal proteins (expressed from the environment gene(s)), and one or more *cells*.
Cell	contains a *genome* and *cytoplasm*, and has some *behaviours*.
Cytoplasm	contains one or more fractal proteins.
Genome	comprising *structural genes* and *regulatory genes*. In this work, the structural genes are divided into different types: *cell receptor genes, environment genes* and *behavioural genes*.
regulatory gene	comprising operator (or promoter) region and coding (or output) region.
cell receptor gene	a structural gene with a coding region which acts like a mask, permitting variable portions of the environmental proteins to enter the corresponding cell cytoplasm.
environment gene	a structural gene which determines which proteins (maternal factors) will be present in the environment of the cell(s).
behavioural gene	structural gene comprising operator and cellular behaviour region.

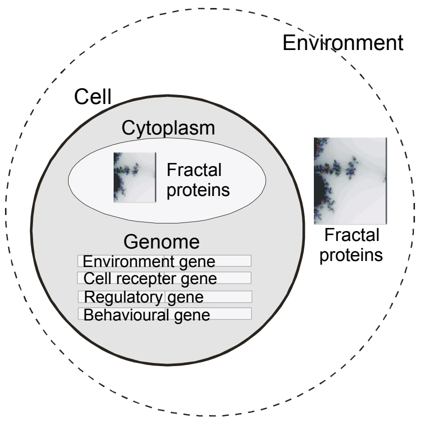

Fig. 1. Representation using fractal proteins.

FRACTAL DEVELOPMENT

```
For every developmental time step:
    For every cell in the embryo:
        Express all environment genes and
        calculate shape of merged environment fractal proteins

        Express cell receptor genes as receptor fractal proteins
        and use each one to mask the merged environment proteins
        into the cell cytoplasm.

        If the merged contents of the cytoplasm match a promoter
        of a regulatory gene, express the coding region of the gene,
        adding the resultant fractal protein to the cytoplasm.

        If the merged contents of the cytoplasm match a promoter of a
        behavioural gene, use coding region of the gene to specify a
        cellular function.

        Update the concentration levels of all proteins in the cytoplasm
        If the concentration level of a protein falls to zero, that protein
        does not exist.
```

Fig. 2. The fractal development algorithm.

In this work, a biologically plausible model of gene regulatory networks is constructed through the use of genes that are expressed into *fractal proteins* – subsets of the Mandelbrot set that can interact and react according to their own fractal chemistry. Further motivations and discussions on fractal proteins are provided in (Bentley, 2004 & 2003a,b,c). Table 1 describes the object types in the representation; Figure 1 illustrates the representation. Figure 2 provides an overview of the algorithm used to develop a phenotype from a genotype. Note how most of the dynamics rely on the interaction of fractal proteins. Evolution is used to design genes that are expressed into fractal proteins with specific shapes, which result in developmental processes with specific dynamics.

3.1 Defining a Fractal Protein

In more detail, a fractal protein is a finite square subset of the Mandelbrot set, defined by three codons (*x,y,z*) that form the coding region of a gene in the genome of a cell. Each (*x, y, z*) triplet is expressed as a protein by calculating the square fractal subset with centre coordinates (*x,y*) and sides of length *z*, see fig. 3 for an example. In addition to shape, each fractal protein represents a certain *concentration* of protein (from 0 meaning "does not exist" to 200 meaning "saturated"), determined by protein production and diffusion rates.

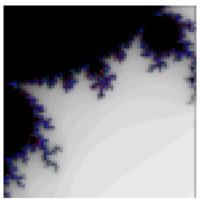

Fig. 3. Example of a fractal protein defined by (*x*=0.132541887, *y*=0.698126164, *z*=0.468306528)

3.2 Fractal Chemistry

Cell cytoplasms and the environment usually contain more than one fractal protein. In an attempt to harness the complexity available from these fractals, multiple proteins are merged. The result is a product of their own "fractal chemistry" which naturally emerges through the fractal interactions.

Fractal proteins are merged (for each point sampled) by iterating through the fractal equation of all proteins in "parallel", and stopping as soon as the length of any is unbounded (i.e. greater than 2). Intuitively, this results in black regions being treated as though they are transparent, and paler regions "winning" over darker regions. See fig 4 for an example.

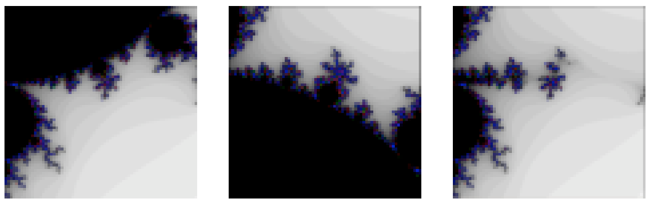

Fig. 4. Two fractal proteins (left and middle) and the resulting merged fractal protein combination (right).

3.3 Genes

The environment gene, cell receptor gene, regulatory genes, and behavioural genes all contain 7 real-coded values:

xp	*yp*	*zp*	*Affinity threshold*	*Concentration threshold*	*x*	*y*	*z*	*type*

where (*xp, yp, zp, Affinity threshold, Concentration threshold*) defines the promoter (operator or precondition) for the gene and (*x,y,z*) defines the coding region of the gene. The *type* value defines which type of gene is being represented, and can be one or all of the following: *environment, receptor*, *behavioural*, or *regulatory*. This enables the type of genes to be set independently of their position in the genome, enabling variable-length genomes. It also enables genes to be multi-functional, i.e. a gene might be expressed both as an environmental protein and a behaviour.

When *Affinity threshold* is a positive value, one or more proteins must match the promoter shape defined by (*xp,yp,zp*) with a difference equal to or lower than *Affinity threshold* for the gene to be activated. When *Affinity threshold* is a negative value, one or more proteins must match the promoter shape defined by (*xp,yp,zp*) with a difference equal to or lower than |*Affinity threshold*| for the gene to be repressed (not activated).

To calculate whether a gene should be activated, all fractal proteins in the cell cytoplasm are merged (including the masked environmental proteins) and the combined fractal mixture is compared to the promoter region of the gene. The full details of this process are beyond the scope of this paper, interested readers should consult (Bentley 2004, 2003a,b and c).

Behavioural Gene. A behavioural gene is activated when other protein(s) in the cytoplasm match its promoter region (using the *affinity threshold*). For this application, a gradual activation between not activated and activated was required, using the *x* value of the coding region (*x,y,z*) triplet as a *fate* value to define a function, calculated as follows:

If the gene is being activated with a negative *Affinity threshold*,
output = *output* - (*totalconcentration* - *concentrationthreshold*) * *fate*

If the gene is being activated with a positive *Affinity threshold*,
output = *output* + (*totalconcentration* - *concentrationthreshold*) * *fate*

Note how the total concentration of proteins seen on the promoter is offset against the *Concentration Threshold* gene and scaled by the *fate* gene (*x* value of the coding region), allowing evolution to adjust the range of values seen on the output, and used to specify behaviours. (If there are more behavioural genes than are required, the resultant behaviour will be the sum behaviour of all genes.)

3.4 Development and Evolution

As was illustrated in figure 2, an individual begins life as a single cell in a given environment. To develop the individual from this zygote into the final phenotype, fractal proteins are iteratively calculated and matched against all genes of the genome. Should any genes be activated, the result of their activation (be it a new protein, receptor or cellular behaviour) is generated at the end of the current cycle. Development continues for *d* cycles, where *d* is dependent on the problem. Note that if one of the cellular behaviours includes the creation of new cells, then development will iterate through all genes of the genome in all cells.

All genes are evolved. The genetic algorithm used in this work has been used extensively elsewhere for other applications (including GADES (Bentley 1999)). A dual population structure is employed, where child solutions are maintained and evaluated, and then inserted into a larger adult population, replacing the least fit. The fittest *n* are randomly picked as parents from the adult population. Typically the child population size is set to 80% of the adult size and $n = 40\%$. (For further details of this GA, refer to (Bentley 1999).) Because real coding was used, duplication and creep mutation is used, see (Bentley 2004) for complete details. Crossover is always applied; all mutations occur with probability 0.01 per gene.

4 Robot Control

Previous work has demonstrated how evolution can generate specific fractal proteins that interact with each other in order to produce a desired robot path through a series of obstacles. The two behavioural genes are used as "steering" and "accelerator" to generate commands for a *wonderborg*™ robot, produced by Bandai, see Fig. 5. Instead of directing cellular behaviour (i.e., cell division, differentiation or death), the fractal gene regulatory networks direct robot behaviour. Full details of the robot and mapping from behavioural genes to robot commands are available in (Bentley 2003a).

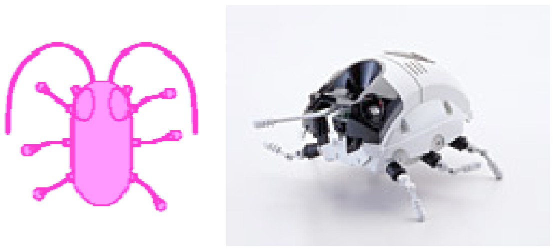

Fig. 5. The *wonderborg*™ robot (Bandai) has six legs, two feelers, two infrared collision detection sensors, a floor sensor, a light sensor and capabilities for the addition of a further motor and sensor.

To enable high-speed evaluation of robot control programs, a *wonderborg*™ simulator was created. This reads the same file format as used by Bandai's proprietary software (and as output by the fractal developmental system) and calculates the path of the robot through an environment. The simulator was designed to be fast – on a 1 Ghz PC, approximately 40 developmental cycles and corresponding robot control simulations occur every second. Further details are available in (Bentley 2003a).

4.1 Environmental Signals

It is not possible to run the fractal GRNs on the robot's simple processor, so in this work the fractal GRNs are used to generate a series of control commands that are then uploaded into the robot. The robot's twin infrared (binary) collision detection sensors were used as inputs, providing four possible states for the robot. (In the language of the *wonderborg*^{TM} these equate to: *nothing there*, *on the left*, *on the right*, or *in front*.) Hence, to generate all necessary control commands, the same GRN was developed four times in succession, in different environments.

As in nature, proteins were used to pass signals from the environment to the genes. Here, the first two environment genes were used to define the fractal proteins produced by the activated sensors, i.e., env. gene 1 defined the protein for sensor 1, and env. gene 2 defined the protein for sensor 2. Other environment genes could be created and used as extra "maternal factors" by evolution. Hence, during the four developmental runs, the relevant environment proteins were switched: "off off" (*nothing there*), "on off" (*on the left*), "off on" (*on the right*) and "on on" (*in front*), representing all possible states of the robot's sensors. The resulting commands produced by the fractal GRN for each input state were interruptible, meaning that should the sensors change, the robot would immediately switch to the commands corresponding to the new input state. With 32 developmental interations, 32 commands (each defining very small moments) were created for each input state. Should the input state remain unchanged after execution of all 32, the robot repeats the same commands.

5 Experiments

The experiments used a basic environment with four obstacles for the robot and an enclosing wall, see figure 6. The robot simulator was initialised with the robot at one end of the environment. The further the robot managed to walk across the environment the higher the fitness of the corresponding controller. If the robot touched an obstacle or an enclosing wall, its final position was measured as its last valid position in the environment. A second fitness measure (of less importance than the first) was used to provide penalties corresponding to the time taken by the robot and hence encourage efficient and fast journeys.

To evolve the controllers, the fractal development system was initialised with a single cell, 2 environment genes, 2 receptor gene, 2 behavioural genes and 6 regulatory genes. (Note that with variable length genomes, evolution was free to modify these gene numbers). The operator and coding regions of the genes were randomly initialised with the alleles that defined 10 previously evolved protein fractals (Bentley, 2004). 32 developmental steps were employed (four times, each in a different environment, see above), and the evolutionary algorithm used a population size of 100, running for up to 100 generations.

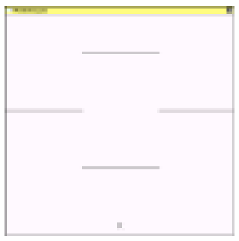

Fig. 6. The environment used for the experiments. The starting position of the robot is marked by "X." The higher the robot manages to walk, the fitter the controller.

6 Results and Analysis

Previous experiments had successfully evolved robot controllers (using no sensors). Typical results were good, but often over a thousand generations were needed to find the intricate path through the obstacles. In this experiment (using sensors), evolution needed typically no more than 5 generations to find impressive robot controllers, capable of following walls and reaching the top of the arena. Collision avoidance and wall-following behaviour evolved in every run, see fig. 7 for an illustration of all twenty controllers.

Analysis shows that most GRNs evolved two different behaviours, for example:

Nothing there / On the right:	Rotate Left 1, Forwards Right 2, Forwards Right 2, … (*31)
In front / On the left:	Rotate Left 1, Rotate Left 1, … (*32)

However, on several occasions (e.g., top right controller of fig. 7), evolution generated three separate behaviours from the same GRN:

Nothing there / On the right:	Forwards left 1, Forwards left 1, … (*32)
On the left:	Forwards right 2, Forwards right 2, … (*32)
In front:	Backup left 1, Backup left 1, … (*32)

The fractal proteins are being used in complex interactions to control the behaviours. Interestingly, changing the environmental proteins hardly seems to alter the pattern of regulatory gene activations within the GRN, but the differences of protein concentrations is enough to change the pattern of behavioural gene activations and cause distinct and different behaviours. It should be noted that the fractal GRNs are not running "natively" on the robot's processor, so previous states of the system cannot influence later states. Should the robot actually calculate fractal GRNs itself, such a concept of memory would be introduced, potentially enabling even better control.

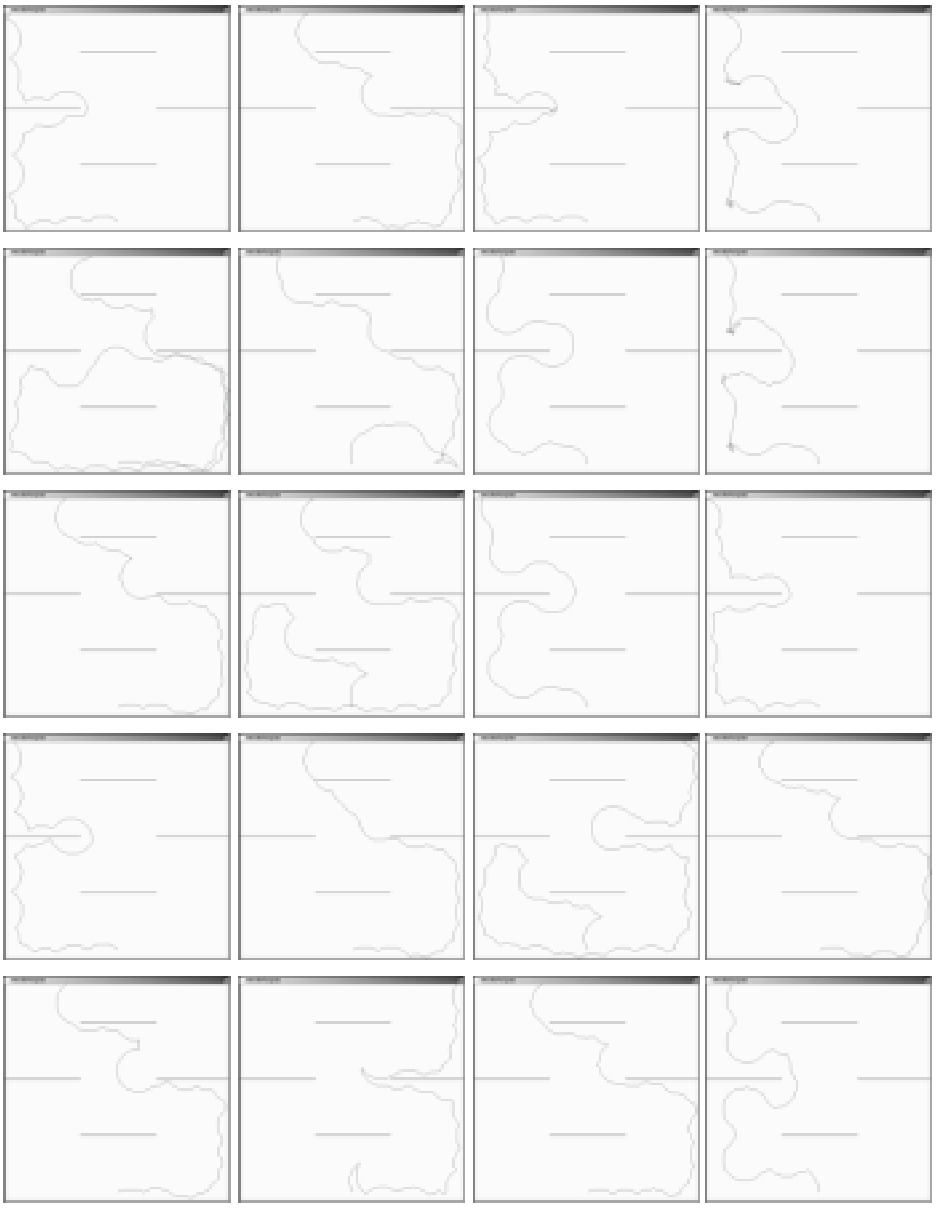

Fig. 7. The twenty evolved robot controllers. All produce general-purpose collision avoidance and wall-following behaviour.

7 Conclusions

Development is an adaptive and highly interactive process, which is capable of generating different solutions in different environments, whether a cellular or organism environment. This work has shown that fractal GRNs can be evolved in different environments to have these properties. Twenty robot controllers were evolved, all demonstrating generic wall-following behaviour. All were evolved in 5 or fewer generations, yet all comprise highly complex interactions between fractal proteins. Future work will investigate the extension of fractal GRNs to multicellular systems.

Acknowledgments

This material is based upon work supported by the European Office of Aerospace Research and Development (EOARD), Airforce Office of Scientific Research, Airforce Research Laboratory, under Contract No. F61775-02-WE014. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of EOARD. MOBIUS is an EMBLEM project.

References

[1] Bentley, P. J. Fractal Proteins. 2004. In Genetic Programming and Evolvable Machines Journal.

[2] Bentley, P. J. Evolving Fractal Gene Regulatory Networks for Robot Control. 2003a. In Proceedings of ECAL 2003.

[3] Bentley, P. J. Evolving Fractal Proteins. 2003b. In Proc. of ICES '03, the 5th International Conference on Evolvable Systems: From Biology to Hardware.

[4] Bentley, P. J. Evolving Beyond Perfection: An Investigation of the Effects of Long-Term Evolution on Fractal Gene Regulatory Networks. 2003c. In Proc of *Information Processing in Cells and Tissues* (IPCAT 2003).

[5] Bentley, P. J. From Coffee Tables to Hospitals: Generic Evolutionary Design. 1999. Chapter 18 in Bentley, P. J. (Ed) Evolutionary Design by Computers. Morgan Kaufmann Pub. San Francisco, pp. 405-423.

[6] Bongard, J. C. Evolving Modular Genetic Regulatory Networks. 2002. In *Proc.of 2002 Congress on Evolutionary Computation*, IEEE Press, pp. 1872-1877.

[7] P. C. Haddow, G. Tufte, and P. van Remortel. Shrinking the Genotype: L-Systems for EHW 2001. In Proc. Of 4th Int. Conf. On Evolvable Systems: From Biology to Hardware, Tokyo, Japan.

[8] Hornby, G. S. Generative Representations for Evolutionary Design Automation. 2003. Brandeis University, Dept. of Computer Science, Ph.D. Dissertation.

[9] A.H. Jackson, A.M. Tyrrell Implementing Asynchronous Embryonic Circuits using AARDVArc. 2002. In *Proceedings of 2002 NASA/DoD Conference on Evolvable Hardware* (EH-2002), IEEE Computing Society, Alexandria, Virginia, pp. 231-240.

[10] N. Jakobi. Harnessing Morphogenesis. 1995. *International Conference on Information Processing in Cells and Tissues*, Liverpool, UK.

[11] S. Kumar and P. J. Bentley. Computational Embryology: Past, Present and Future. 2003. Invited chapter in Ghosh and Tsutsui (Eds) Theory and Application of Evolutionary Computation: Recent Trends. Springer Verlag (UK).

[12] Mandelbrot, B. *The Fractal Geometry of Nature*. 1982. W.H. Freeman & Company.

[13] Miller, J. and Banzhaf, W. Evolving the Program for a Cell: From French Flags to Boolean Circuits. 2003. Invited chapter in Kumar, S. and Bentley, P. J. (Eds) *On Growth, Form and Computers*. Academic Press, 2003.

[14] Quick, T. Evolving Embodied Genetic Regulatory Network-Driven Control Systems. 2003. In Proc. of ECAL 2003.pp. 266-277.

[15] Lewis Wolpert, Rosa Beddington, Thomas Jessell, Peter Lawrence, Elliot Meyerowitz, Jim Smith. *Principles of Development, 2nd Ed.* 2001. Oxford University Press.