Biologically Inspired Evolutionary Development

Sanjeev Kumar and Peter J. Bentley

Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
S.Kumar@cs.ucl.ac.uk P.Bentley@cs.ucl.ac.uk

Abstract. We describe the combination of a novel, biologically plausible model of development with a genetic algorithm. The Evolutionary Developmental System is an object-oriented model comprising proteins, genes and cells. The system permits intricate genomic regulatory networks to form and can evolve spherical embryos constructed from balls of cells. By attempting to duplicate many of the intricacies of natural development, and through experiments such as the ones outlined here, we anticipate that we will help to discover the key components of development and their potential for computer science.

1 Introduction

Talk to any evolutionary biologist and they'll tell you that the standard genetic algorithm (GA) does not resemble natural evolution very closely. While our GAs may evolve their binary genes, most biologists would be horrified to discover that concepts such as genotype and phenotype are so blurred in evolutionary computation that some researchers make no distinction between the two. Should you have the courage to go and talk to a developmental biologist, you'll have an even worse ear-bashing. You'll be told that development is the key to complex life. Without a developmental stage from genotype to phenotype, all you have is a big DNA or RNA molecule. With development you can have layer upon layer of complexity, from cells to organs to organisms to societies.

Of course our motivations in computer science are often very different from the motivations of biologists. Nevertheless, it has long been the goal of evolutionary computationists to evolve complex solutions to problems without needing to program-in most of the solution first. The dream of complex technology that can design itself requires rejection of the idea of knowledge-rich systems where human designers dictate what should and should not be possible. In their place we need systems capable of building up complexity from a set of low-level components. Such systems need to be able to learn and adapt in order to discover the most effective ways of assembling components into novel solutions. And this is exactly what developmental processes in biology do, to great effect.

In this paper we present, for the first time, an overview of a novel biologically plausible model of development for evolutionary design. This system is intended to model biological development very closely in order to discover the key components

of development and their potential for computer science. The paper is divided into sub-sections covering different aspects of the Evolutionary Developmental System (EDS). It begins with an overview of the entire system, followed by sections detailing individual components in isolation. These individual components are then drawn together, and how they work as part of the overall developmental system is detailed as well as the role of evolution and how the genetic algorithm is wrapped around the developmental core. Finally we present some examples of results generated during on-going experiments.

2 Background

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 [Bentley, 2002].

The field of Computational Development has matured steadily over the past decade or so, with work touching upon a wide range of aspects of development ranging from its use for the construction of neural net robot controllers [Jakobi, 1996], to the large scale modelling of morphogenesis [Fleischer, 1993].

Recently a resurgence of interest into computational development has fuelled much research. Problems of scalability, adaptability and evolvability have led many researchers to attempt to include processes such as growth, morphogenesis or differentiation in their evolutionary systems [Eggenberger, 1996; Haddow et al, 2001; Bongard, 2002; Miller 2002]. For reviews see [Kodjabachian and Meyer, 1994; Kumar and Bentley, 2002].

3 The Evolutionary Developmental System (EDS)

In nature, development begins with a single cell: the fertilised egg, or zygote. In addition to receiving genetic material from its two parents, the zygote is seeded with a set of proteins — the so-called 'maternal factors' deposited in the egg by the mother [Wolpert, 1998]. The maternal factors trigger development causing the zygote to cleave (fast cell division with no cell growth). After cleavage, normal cell division begins; as cells divide they inherit the state of their parents. To ensure the embryo is not homogenous, one or two asymmetric divisions occur, resulting in an unequal distribution of factors to the daughter cells. In doing so, cells become different from one another.

Development is controlled by our DNA. In response to proteins, genes will be expressed or repressed, resulting in the production of more (or fewer) proteins. The chain-reaction of activation and suppression both within the cell and within other

nearby cells through signaling proteins and cell receptors, causes the complex processes of cellular differentiation, pattern formation, morphogenesis and growth.

The Evolutionary Developmental System is an attempt to encapsulate many of these processes within a computer model. At the heart of the EDS lies the developmental core. This implements concepts such as embryos, cells, cell cytoplasm, cell wall, proteins, receptors, transcription factors (TFs), genes, and cis-regulatory regions (see figure 1 for a graphical view of the EDS). Genes and proteins form the atomic elements of the system. A cell stores proteins within its cytoplasm and its genome (which comprises rules that collectively define the developmental program) in the nucleus. The overall embryo is the entire collection of cells (and proteins emitted by them) in some final conformation attained after a period of development. A genetic algorithm is wrapped around the developmental core. This provides the system with the ability to evolve genomes for the developmental machinery to execute.

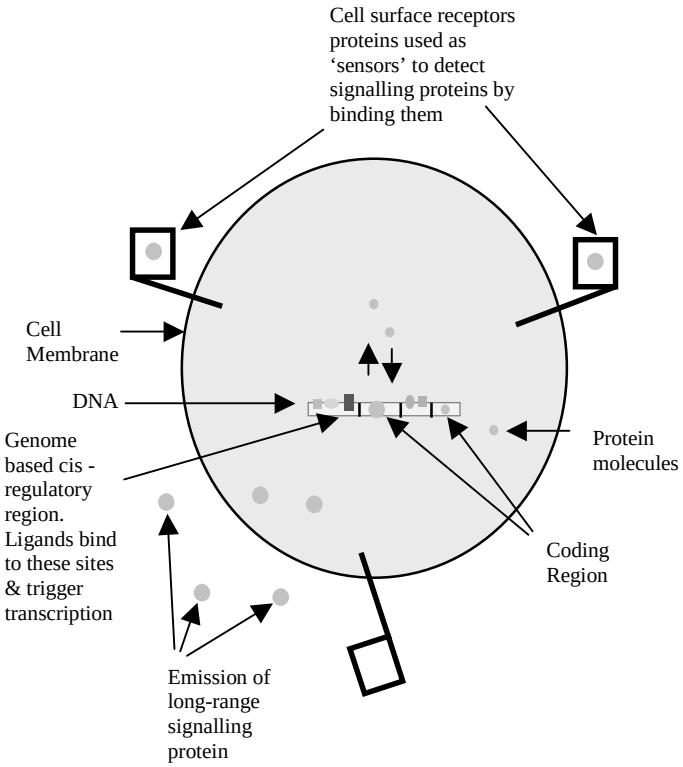

Figure 1: A single cell in the Evolutionary Developmental System

4 Components of the EDS

The following sections describe the main components of the developmental model: proteins, genes and cells.

4.1 Proteins

In nature, proteins are the driving force of development. They are macromolecules, long chains of amino acids that assemble at protein production sites known as ribosomes. The only function of genes is to specify proteins.

The EDS captures the concept of a protein as an object. Each protein has an ID tag, which is simply an integer number. The EDS uses eight proteins (although number of proteins used is a user defined variable in the system). Protein objects contain both a current and a new state object (at the end of each developmental cycle all protein new states are swapped with current states to provide "parallel" protein behaviour). These protein state objects house important protein-specific information, for example, the protein diffusion co-efficient.

Protein creation, initialisation, and destruction. In the EDS, proteins do not exist in isolation; they are created and owned by cells. Thus, during protein construction each protein is allocated spatial co-ordinates inherited from the cell creating the protein. Handling protein co-ordinate initialisation using this method overcomes the problem of knowing which cell created which proteins.

A protein lookup table (extracted from the genome, see next section) holds details about all proteins and is used to initialise each protein upon creation. It has the following details for each protein:

Rate of Synthesis	amount by which the protein is synthesised
Rate of Decay	amount by which the protein decays
Diffusion coefficient	amount by which the protein diffuses
Interaction strength	strength of protein interaction, i.e., activation or inhibition
Protein Type	ID tag, e.g., long-range hormone, or short-range receptors

Additionally, each protein keeps the following variables:

Bound?	whether or not a receptor protein is currently bound[1]
Protein Source Concentration	the current concentration of the protein
Spatial coordinates	the position of the source of the protein

Protein destruction in the EDS is implemented by simply setting the protein's source concentration to zero: if the concentration is zero there can be no diffusion, unless more of the protein is synthesised.

[1]The bound variable is only operational in receptor proteins.

Protein Diffusion. Diffusion is the process by which molecules spread or wander due to thermal motions [Alberts et al., 1994]. When molecules in liquids collide, the result is random movement. Protein molecules are no different: they diffuse.

The average distance that a molecule travels from its starting point is proportional to the square root of the time taken to do so. For example, if a molecule takes on average 1 second to move 1 μm, it will take 4 seconds to move 2 μm, 9 seconds to move 3 μm, and 100 seconds to move 10 μm. Diffusion represents an efficient method for molecules to move short distances, but an inefficient method to move over large distances. Generally, small molecules move faster than large molecules (Alberts et al., 1994).

Protein diffusion in the EDS models this behaviour. Diffusion is implemented by using a Gaussian function centred on the protein source. The use of the Gaussian assumes proteins diffuse equally in all directions from the cell.

In more detail: the source concentration records the amount of the current protein. Every iteration, its value is decremented by the corresponding 'rate of decay' parameter. If expressed by a gene, its value is also incremented by the corresponding 'rate of synthesis' parameter. To calculate the concentration of a protein at a distance x from the protein source:

$$concentration = s \times \mathbf{e}^{\frac{-x^2}{2d^2}}$$

Where: d is the diffusion coefficient of the current protein.
x is distance from protein source to current point
s is the current protein source concentration.

Figure 2 illustrates the way protein concentration changes according to the three variables: distance, diffusion coefficient and source concentration.

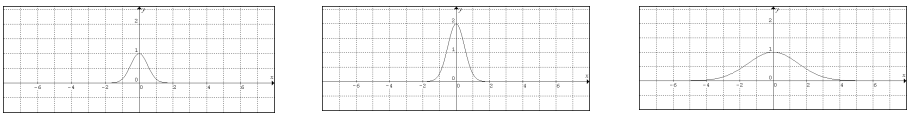

Figure 2. Plot of protein concentration against distance from source, where: d = 0.5 and s = 1.0 (left), d = 0.5 and s = 2.0 (middle), and d = 1.5 and s = 1.0 (right).

4.2 Genes

The EDS employs two genomes. The first contains protein specific values (e.g., synthesis, decay, diffusion rates, see above). These are encoded as real floating-point numbers. The second describes the architecture of the genome to be used for development; it describes which proteins are to play a part in the regulation of different genes. It is this second genome that is employed by each cell for development; the information evolved on the first genome is only needed to initialise proteins with their respective properties.

In Nature, genes can be viewed as comprising two main regions: the cis-regulatory region [Davidson, 2001] and the coding region. Cis-regulatory regions are located just before (upstream of) their associated coding regions and effectively serve as switches that integrate signals received (in the form of proteins) from both the extra-cellular environment and the cytoplasm. Coding regions specify a protein to be transcribed upon successful occupation of the cis-regulatory region by assembling transcription machinery. Currently, the EDS's underlying genetic model assumes a "one gene, one protein" simplification rule (despite biology's ability to construct multiple proteins); this aids in the analysis of resulting genetic regulatory networks. To this end, the activation of a single gene in the EDS results in the transcription of a single protein. This is currently ensured by imposing the following structure over genes: each gene comprises both a cis-regulatory region and a consequent protein-coding region.

A novel genome representation (based on eukaryotic genetics) was devised for development in the EDS. This genome is represented as an array of Gene objects (fig. 3). Genes are objects containing two members: a cis-regulatory region and a protein-coding region. The cis-regulatory region contains an array of TF target sites; these sites bind TFs in order to regulate the activity of the gene.

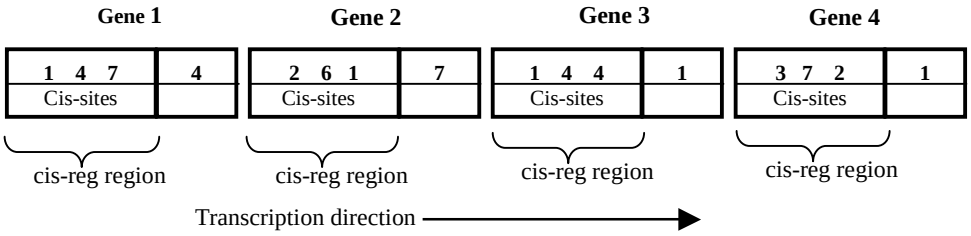

Figure 3. An arbitrary genome created by hand. Genes consist of two objects: a cis-regulatory region and a coding region. Cis-regulatory regions consist of transcription factor target sites that bind TFs, triggering transcription of the coding region. Each number denotes a protein.

The gene then integrates these TFs and either switches the gene 'on' or 'off'. Integration is performed by summing the products of the concentration and interaction strength (weight) of each TF, to find the total activity of all TFs occupying a single gene's cis-regulatory region:

$$a = \sum_{i=1}^{d} conc_i * \text{int}eraction\_strength_i$$

where: a is the total activity, i is the current TF,
d is the total number of TF proteins visible to the current gene,
$conc_i$ is the concentration of i at the centre of the current cell,
$interaction\_strength_i$ is the strength of protein interaction for the current TF (see previous section).

This sum provides the input to a logistic sigmoid threshold function (a hyperbolic tangent function), which yields a value between –1 and 1. Negative values denote gene repression and positive values denote gene activation:

$$g(a) \equiv \tanh(a) \equiv \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

Figure 4 illustrates this sigmoid calculation used to determine whether a gene is activated and produces its corresponding transcription factor or not.

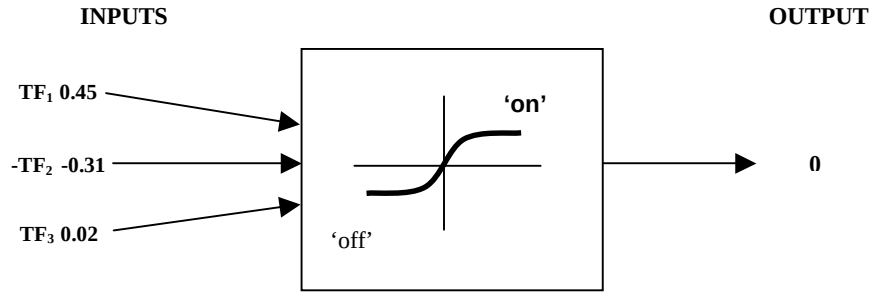

Figure 4. A gene showing the various positive and negative inputs received in the form of transcription factors, with their respective affinities (weights), and concentrations of 0.24, 0.87, and 0.11 respectively. Internally, the gene integrates these TFs and decides whether or not to switch the gene 'on' or 'off'. TF_1 and TF_3 are both activators, whereas TF_2 is a repressor, denoted by a '-' symbol.

4.3 Cells

Cells can be viewed as autonomous agents. These agents have sensors in the form of surface receptors able to detect the presence of certain molecules within the environment. Additionally, the cell has effectors in the form of hundreds and thousands of protein molecules transcribed from a single chromosome able to affect other genes in other cells. Cells resemble multitasking agents, able to carry out a range of behaviours. For example, cells are able to multiply, differentiate, and die.

Like protein objects, cell objects in the EDS have two states: *current* and *new*. During development, the system examines the current state of each cell, depositing the results of the protein interactions on the cell's genome in that time step into the new state of the cell. After each developmental cycle, the current and new state of each cell is swapped ready for the next cycle.

The EDS supports a range of different cell behaviours, triggered by the expression of certain genes. These are currently: division (when an existing cell "divides", a new cell object is created and placed in a neighbouring position), differentiation (where the function of a cell is fixed, e.g., colour = "red" or colour = "blue"), and apoptosis (programmed cell death).

The EDS uses an *n-ary* tree data structure to store the cells of the embryo, the root of which is the zygote. As development proceeds, cell multiplication occurs. The resulting cells are stored as child nodes of the root in the tree. Proteins are stored within each cell. When a cell needs to examine its local environment to determine which signals it is receiving, it traverses the tree, checks the state of the proteins in each cell against its own and integrates the information.

4.4 Evolution

A genetic algorithm (GA) is "wrapped around" the developmental model. The GA represents the driving force of the system. Its main roles are to:

1. provide genotypes for development;
2. provide a task or function, and hence a measure of success and failure; and
3. search the space of genotypes that give rise to developmental programs capable of specifying embryos, correctly and accurately according to the task or function.

Individuals within the population of the genetic algorithm comprise a genotype, a phenotype (in the form of an embryo object), and a fitness score. After the population is created, each individual has its fitness assessed through the process of development. Each individual is permitted to execute its developmental program according to the instructions in the genome. After development has ended a fitness score is assigned to the individual based upon the desired objective function.

The EDS uses a generational GA with tournament selection (typically using ¼ of population size), and real coding. Crossover is applied with 100% probability. Creep mutation is applied with a Gaussian distribution (small changes more likely than large changes), with probability between 0.01 and 0.001 per gene.

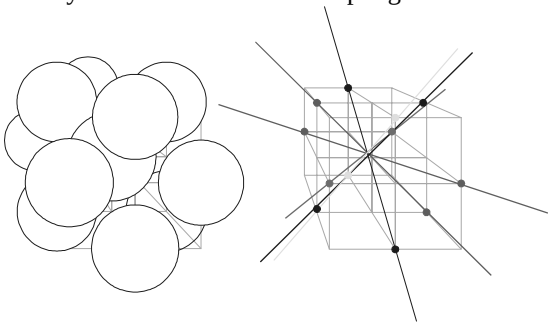

Figure 5. Isopatial coordinates permit twelve equidistant neighbours for each cell (left) and are plotted using six axis (right).

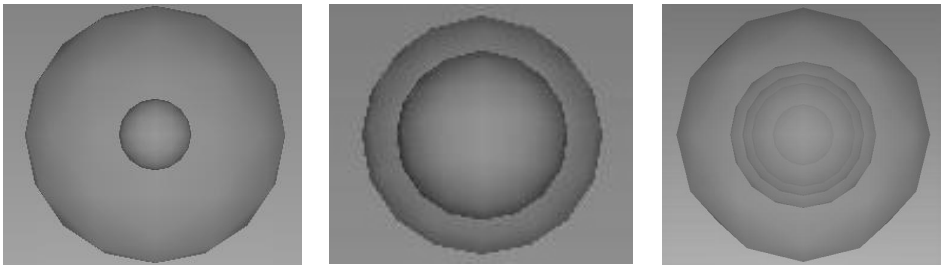

Figure 6. Examples of proteins with their associated cell (at centre). Left: single cell emitting a long-range hormone-type protein. Middle: single cell emitting a short range (local) protein. Right: single cell emitting four proteins of various spread, reflected by the radius of each protein sphere.

4.6 Coordinates and Visualisation

The underlying co-ordinate system used by the EDS is isospatial. All coordinate systems have inherent biases towards different morphologies; the isospatial system is no different. However, the isospatial system bias results in what can only be described as more natural (biologic) morphologies than its Cartesian counterpart [Frazer, 1995]. Isospatial co-ordinates permit a single cell to have up to twelve equidistant neighbours defined by 6 axis (fig. 5), Cartesian co-ordinates only permit 6.

The EDS automatically writes VRML files of developed embryos, enabling three-dimensional rendered cells and proteins to be visualised. Cells are represented by spheres of fixed radius; proteins are shown as translucent spheres of radius equal to the extent of their diffusion from their source cells. In order to place a cell in VRML its Cartesian co-ordinates need to be defined: to this end, isospatial co-ordinates are converted to Cartesian. Figure 6 illustrates how cells and proteins appear when rendered.

5 Experiments

Because of the complexity of the system, numerous experiments can be performed to assess behaviour and capabilities. Here (for reasons of space) we briefly outline two:

1. The ability of genes and proteins to interact and form genomic regulatory networks within a single cell.
2. The evolution of a 3D multi-cellular embryo with form as close to a prespecified shape as possible.

5.1 Genetic Regulatory Networks

In order to assess the natural capability of the EDS to form GRNs independently of evolution, genomes of five random genes were created and allowed to develop in the system for ten developmental steps. The cell was seeded with a random set of eight proteins (maternal factors).

Figure 7 (top) shows an example of the results of this experiment. The pattern shows gene four exhibiting autocatalytic behaviour having initially bound to protein zero. (Gene four is activated when in the presence of protein zero, and produces protein zero when activated.)

Figure 7 (bottom) shows an example of the pattern that results when the initial random proteins (initial conditions) are varied very slightly, but the genome is kept constant. Again, gene four shows the same autocatalytic behaviour, but now the GRN has found an alternative pattern of activation. These two runs illustrate the difference the initial proteins can make on the resulting GRN.

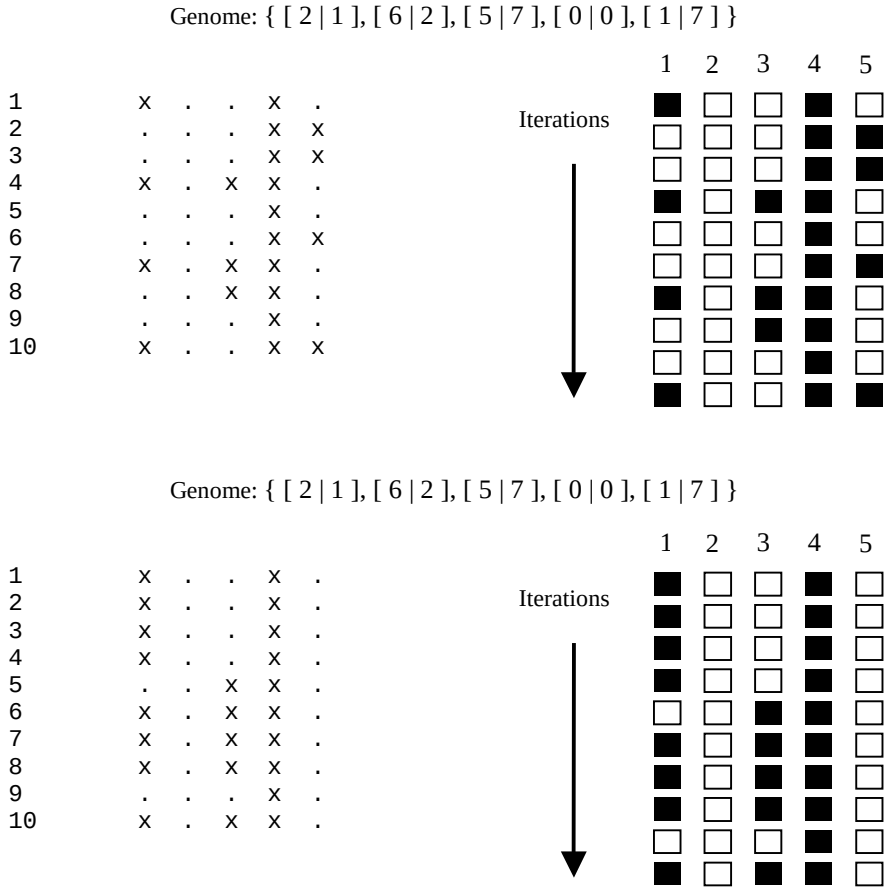

Figure 7. Gene expression patterns for a run of a randomly created genome seeded with a random subset of proteins. The left side shows the raw output from the system where an 'x' means the gene in that column is 'on' and '.' means the gene is 'off'. The right side depicts this text pattern as a graphical output viewed as a 1D CA iterated over ten time-steps. Note, gene 4, i.e., [0 | 0] is autocatalytic.

5.2 Morphogenesis: Evolving a Spherical Embryo

In addition to GRNs, the other important capability of the EDS is cellular behaviour. The second experiment focuses on morphogenesis, i.e., the generation of an embryo with specific form, constructed through appropriate cellular division and placement, from an initial single zygote. For this experiment, the genetic algorithm was set up as described previously, with the fitness function providing selection pressure towards spherical embryos of radius 2 (cells have a radius of 0.5).

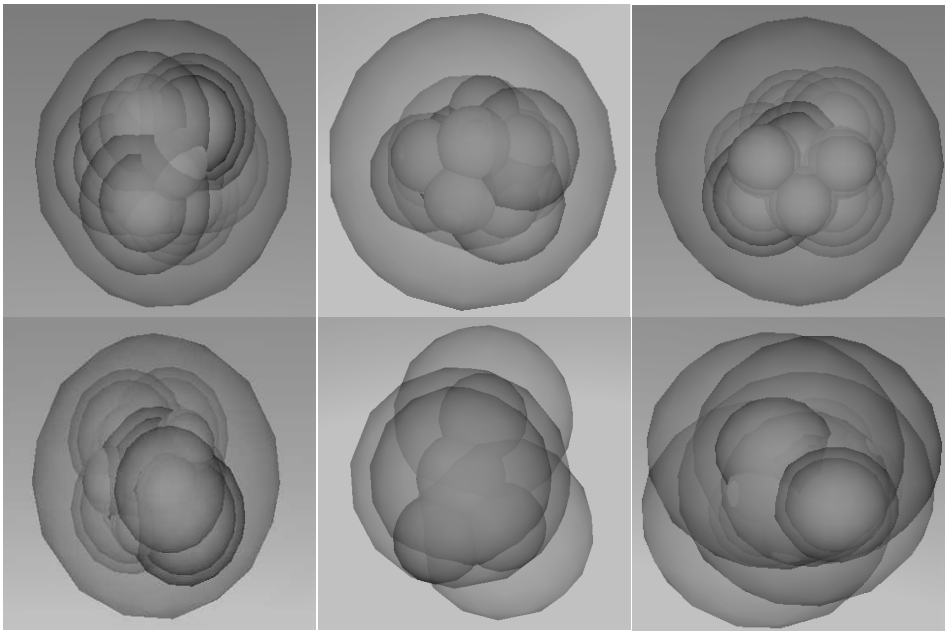

Figure 8. Six random initial embryos.

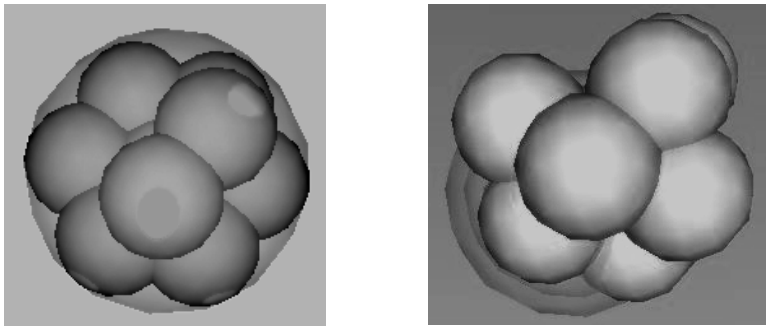

Figure 9. Two "spherical" embryos. Using the equation of a sphere as a fitness function with sphere of radius 2.0.

Figure 8 shows examples of the initially random embryos with their corresponding proteins produced by the GRNs. Figure 9 shows two examples of final "spherical" embryos. As well as having appropriate forms, it is clear that the use of proteins has been reduced by evolution. Interestingly, analysis indicates that evolution did not require complex GRNs to produce such shapes. It seems likely that it is the natural tendency of the EDS to produce near-spherical balls of cells, hence evolution simply did not need to evolve intricate GRNs for this task. Further experiments to evolve more complex morphologies are under way.

6 Summary

The staggering complexities of nature result from a combination of evolution and development. This work has described the combination of a novel, biologically plausible model of development with a genetic algorithm. We have shown how an Evolutionary Developmental System can be constructed based on an object-oriented model of proteins, genes and cells. We have also described how this system permits intricate genomic regulatory networks to form and can evolve spherical embryos constructed from balls of cells. By attempting to duplicate many of the intricacies of natural development, and through experiments such as the ones outlined here, we anticipate that we will help to discover the key components of development and their potential for computer science. Further experiments and analysis are ongoing.

Acknowledgements

Many thanks to Lewis Wolpert and Michel Kerszberg for helpful advice and criticism. Thanks also to Tom Quick and Piet van Remortel for helpful suggestions.

References

1 Alberts et al. (1994) *Molecular Biology of the Cell*. 3rd edition. Garland Publishing.
2 P. J. Bentley (2002) *Digital Biology*. Simon and Schuster, New York.
3 J. Bongard (2002) Evolving Modular Genetic Regulatory Networks. In *Proceedings of the IEEE 2002 Congress on Evolutionary Computation*.
4 E.H. Davidson (2001). *Genomic Regulatory Systems*. Academic Press.
5 P. Eggenberger (1996) Cell interactions as a control tool of developmental processes for evolutionary robotics. In Maes, P. et al.(Eds) *From Animals to Animats 4*. Cambridge, MA: MIT Press.
6 K. Fleischer and A. Barr (1993). A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In C. Langton, editor, *Artificial life III*, pages 389--416. Addison-Wesley, 1993.
7 J. Kodjabachian and J.-A. Meyer (1994). Development, learning and evolution in animats. In *Perception To Action Conference Proceedings*, P. Gaussier and J.-D. Nicoud, Eds. 1994, pp. 96--109, IEEE Computer Society Press.
8 J. Frazer (1995). *An Evolutionary Architecture.* Architecture Association, London.
9 P. C. Haddow, G. Tufte, and P. van Remortel (2001) "Shrinking the Genotype: L-Systems for EHW?" In Proc. Of 4th Int. Conf. On Evolvable Systems: From Biology to Hardware, Tokyo, Japan.
10 N. Jakobi (1996c). Harnessing morphogenesis. In Proceedings of *the International Conference on Information Processing in Cell and Tissue.*
11 S. Kumar and P. J. Bentley (2002). Computational Embryology: Past, Present and Future. Invited chapter in Ghosh and Tsutsui (Eds) Theory and Application of Evolutionary Computation: Recent Trends. Springer Verlag (UK).
12 J. F. Miller (2002) "What is a Good Genotype-Phenotype Mapping for the Evolution of Computer Programs?" Presented at the *Software Evolution and Evolutionary Computation Symposium*, EPSRC Network on Evolvability in Biology & Software Systems, University of Hertfordshire, Hatfield, U.K.7-8 February 2002.
13 L. Wolpert (1998), *Principles of Development*, Oxford University Press.