Harnessing Morphogenesis

CSRP 423

Nick Jakobi
School of Cognitive and Computing Sciences
University of Sussex, Brighton BN1 9QH, England
email: nickja@cogs.susx.ac.uk

Abstract

This paper explains in detail a biologically inspired encoding scheme for the artificial evolution of neural network robot controllers. Under the scheme, an individual cell divides and moves, in response to protein interactions with an artificial genome, to form a multi-cellular 'organism'. After differentiation dendrites grow out of each cell, guided by chemically sensitive growth cones, to form connections between the cells. The resultant network is then interpreted as a recurrent neural network robot controller. Results are given of preliminary experiments to evolve robot controllers for both corridor following and object avoidance tasks.

Keywords: Artificial Evolution, Morphogenesis, Evolutionary Robotics.

1 Introduction

Many people have noted the difficulties involved in designing control architectures for robots by hand ([10], [1]) and as robots and the behaviours we demand of them become more complicated these difficulties can only increase. Evolutionary robotics is an attempt to overcome some of these difficulties by automating the design process (see [9] for an example). Artificial evolution is employed to evolve control architectures using a genetic algorithm [4].

In the experiments reported in this paper, a population of initially random genotypes (character strings) are developed into phenotypes (artificial neural networks) and assigned a fitness value based on their ability to control a small mobile robot at a simple task. A new generation of genotypes is then 'bred' from the old population by probabilistically selecting fit parent genotypes and applying the genetic operators of crossover and mutation to create offspring genotypes. Over many generations, average fitness gradually improves until individuals evolve that can perform the task satisfactorily.

Both developmental biology and evolutionary robotics, then, deal with processes by which genotypes become phenotypes. This is, at the moment, where the similarity between the two disciplines ends. With a few notable exceptions (amongst them [2], [14] and [16]), the crude bit-string to neural network transformations normally used in evolutionary robotics are clumsy parodies of the subtle mechanisms of Morphogenesis that developmental biologists are only beginning to uncover. In itself this is no bad thing, but since no entirely satisfactory method of encoding really complicated neural networks in a compact way has been discovered, this paper represents a step towards exploiting the encoding scheme used by mother nature.

The inspiration behind the encoding scheme described in this paper comes from biology, but it was designed with evolutionary robotics firmly in mind, and not as a defensible model of biological development. The overall picture to be gleaned from the model, described in the next section, will at least be reminiscent to the developmental biologist of what actually takes place during Morphogenesis, but where precise functional details of a particular developmental mechanism were unavailable, or where the biological details appeared to conflict with the goals of evolvability and flexibility of design necessary to any encoding scheme, alternative mechanisms were implemented. Biological terms are used extensively throughout the paper. Unless stated otherwise they should be seen as explanatory tools rather than strong references to their biological counterparts.

2 An Overview of the Developmental Model

This section gives a general overview of the encoding scheme in three stages. First it describes what takes place at the level of the genome, then how this controls the behaviour of individual cells, and finally how a multicellular 'organism' develops from a single cell.

In the current scheme, a genome consists of a single string of what might be thought of as base-pairs of nucleic acids (i.e. one of four characters). The start of each gene on the genome is identified by a certain pattern of preceding characters, similar to the TATA box on a real genome (see [6] for a good introduction to developmental biology). Genes are of fixed length. Each gene is responsible for the production of a particular protein and in turn is regulated by certain combinations of proteins within the cell. What is important to realise at this stage is that genes create proteins that regulate genes (see [15] for a beautifully clear example of this), forming genomic regulatory networks (GRNs) that control the entire behaviour of each cell during development. These are best thought of as independent dynamical systems within each cell, capable of being knocked from one basin of attraction into another by internal and external stimuli, following independent trajectories through state space [1] (for full details see Section 4.1).

The proteins within each cell are divided into different classes. Each has a unique effect on the gross behaviour of the cell. The ability of a given protein to perform its role within its class depends on its 'shape' (for a full explanation of each class and how they interact with each other see Section 4.3). Signal proteins diffuse out of one cell and into another, allowing cells to influence each other at a distance. The *direct* consequences of this influence may take one of two forms, either perturbing the cell's internal dynamics (turning certain genes 'on' and others 'off') or applying a force on the cell body towards or away from the protein source. In practice, there are many different signal proteins, from many different sources, entering each cell at any given time, and each cell will only respond to a subset (depending on the state

[1] For a fuller account of this interpretation and, to my knowledge, the first exposition of genomic functionality as a regulatory network, see [13].

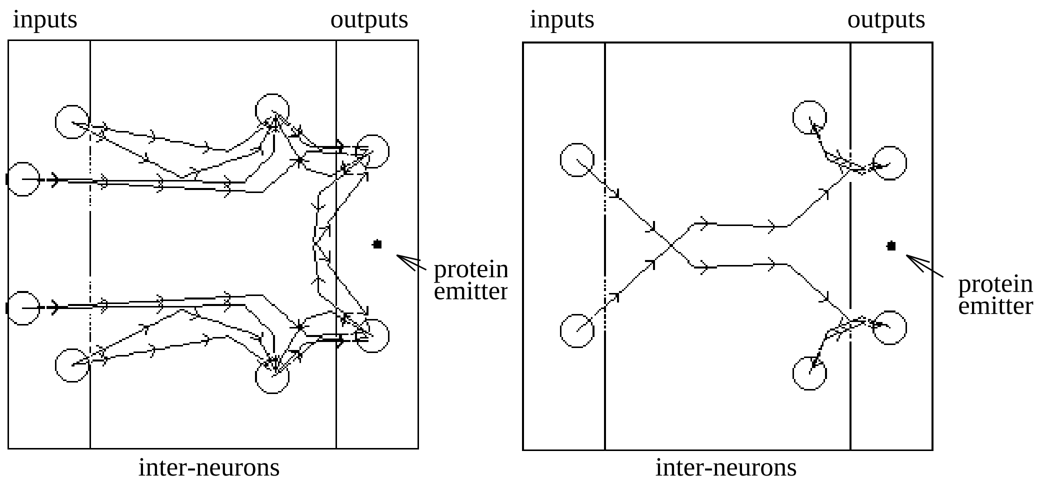

Figure 1: *Two evolved neural networks. The network on the left exhibits corridor following behaviour, while that on the right exhibits obstacle avoiding behaviour. The input region of the developmental environment was divided into eight sub-regions, from top to bottom, corresponding to the eight infra red light sensors on the Khepera robot. The output region was divided into two sub-regions, corresponding to the left and right motors of the Khepera.* small

of its internal dynamics). This means that different cells may behave differently in response to identical external stimuli, and also that different cells may behave identically in response to different external stimuli.

Initially a single cell is placed in an environment containing a number of strategically placed extra-cellular signal protein sources. These fixed sources provide a reference for the developmental process. After the initial cell begins to divide, individual cells divide and move, as part of one big dynamical system coupled together by signal protein interactions, until, eventually, they differentiate (see section 4.4). When this occurs, a number of dendrites grow out from each cell guided by 'growth cones' sensitive to unique combinations of signal proteins. On contact with another cell, a synaptic connection is formed (see Section 4.5). After every cell has differentiated and every dendrite has either connected or died, thresholds and weights are assigned to each cell and dendrite respectively, and inputs and outputs are assigned to cells which lie in specific regions of the developmental environment, to form a neural network ready for testing (see section 4.6).

3 Preliminary Experiments

Before going into the details of the encoding scheme, I will first outline the results of preliminary experiments. This is done so that the reader may understand the general ideas behind the work reported here before becoming too bogged down with the finer points.

Two initial experiments were carries out to test the encoding scheme. Both were carried out using a simulation [11] of the Khepera miniature mobile robot [12] of the the correct level of detail to ensure transfer of evolved behaviours from simulation to robot. Both experiments involved a population of one hundred character strings, each of which was 5000 (initially random) characters long. The genetic algorithm in both cases was based on a simple generational model with rank-based selection. Crossover and mutation were the only genetic operators used. Crossover happened at every 'breeding' and the mutation rate was set at 0.002 mutations per character on the genome (about 12 mutations per genome on average).

In the first experiment, the robot was placed at one end of a long corridor with many bends, and neural networks were evolved that could guide it down the corridor without crashing into the walls. The fitness function (similar to that used in [3]) returned the normalised product of three terms: one for going as fast as possible, one for going as straight as possible and one for staying away from walls. By generation 10 networks had evolved that could guide the robot down the corridor. The network displayed in Figure 1, which was the fittest individual of generation 50, could successfully guide the robot down the corridor without it touching the walls. The experiment was run for a total of 500 generations but no real improvement was made after the fiftieth generation.

In the second experiment, the robot was placed in a rectangular environment containing many small cylinders. The fitness function was the same as that used above. Again, by generation 10, neural networks had evolved which evoked obstacle avoiding behaviour in the robot. The network displayed on the right hand side of Figure 1 is again the fittest member of the 50th generation. This network proved to be very good at obstacle avoidance. However, since it only makes use of two of the available eight sensors on the robot, the robot has several 'blind spots', including straight ahead for small objects.

4 The Encoding Scheme in Detail

The reader who is not interested in the details of the algorithm that converts character strings to artificial neural networks may skip this section if they want. They should, however, be warned that a full appreciation of the issues raised in Section 5 is not possible without a more involved understanding of the encoding scheme.

4.1 The Genomic Regulatory Network (GRN)

Proteins regulate genes which produce proteins. In other words proteins regulate other proteins. Each unit in the GRN corresponds to exactly one protein in the cell and the pattern and nature of links between units is defined by which protein(s) regulate which other protein(s). The activity of each unit equals the intra-cellular concentration of its particular protein (which is equal in turn to the activity of the gene that produces it: maximum 1.0, minimum 0.0). In reality, it may (and usually does) take the *presence* of several proteins and the *absence* of several others to activate a particular gene (thus increasing the cellular concentration of the protein it encodes for). There are usually several links fanning into each unit in the GRN, each with a weight between 2.0 and -2.0, and each unit has a unique threshold that sets the lower bound of the linear threshold activation function.

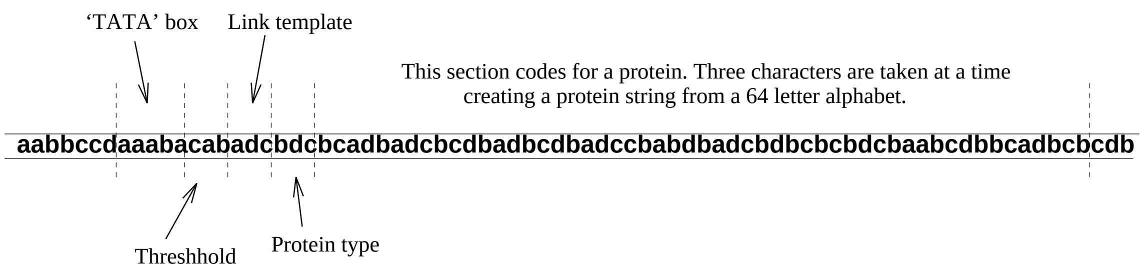
'TATA' box
Link template
This section codes for a protein. Three characters are taken at a time creating a protein string from a 64 letter alphabet.
aabbccdaaabacabadcbdcbcadbadcbcdbadbcdbadccbabdbadcbdbcbcbdcbaabcdbbcadbcbcdb
Threshhold
Protein type

Figure 2: *The lay-out of a gene.*

4.2 Preprocessing The Genome

Various forms of template matching routines are used to work out the nature of the GRN and indeed all protein interactions, including those that have no direct influence on genomic activity. These routines are computationally expensive but, since the GRN and all protein interactions remain fixed for any given genome, the genome may be exhaustively preprocessed before development begins, and then discarded. This leaves the GRN (which may now be treated exclusively as a recurrent neural network with linear threshold activation functions[2]), and various relations between the units of the GRN, extra-cellular stimuli and internal variables that have a direct bearing on the gross behaviour of the cell.

The makeup of a gene is shown in Figure 2. As with a real gene there is a short regulatory region that contains several fields and a longer coding region. The 'TATA' region marks the start of the gene, the threshold region defines the amount of stimulation needed to switch the gene on, the link template controls which proteins affect the gene's activity and the type region defines what class of protein the gene codes for. A protein is translated from the coding region by taking triplets of characters to produce a string whose characters derive from a 64 letter alphabet. The ends of each protein string are then joined together to create a circle.

There are two types of template matching, protein-to-genome and protein-to-protein. These routines are used in three different ways: to find relationships between proteins and the genome, to find relationships between proteins and other proteins, and to find relationships between proteins and fixed class-specific templates (designed to elicit specific information about the functionality of that protein within its class). Each of the 64 characters that may occur in a circular protein string can be regarded as a chemical with a unique affinity for each of the four chemicals (characters) that make up the genome and with a unique affinity for each of the other 63 chemicals that may constitute a protein. All affinities are values between 0 and 1. Any single character match (protein-to-protein or protein-to-genome) will, therefore, result in one of 64 values between zero and one. These routines, in the current implementation, usually operate on templates of three contiguous characters. These matches return one of 192 values between 0 and 2.

The lower bound of the linear threshold activation function of each unit in the GRN is read directly from the threshold region of the corresponding protein's gene. The triplet of characters is converted from base 4 to a decimal number between 0

[2]Care should be taken to distinguish the GRN from the phenotype as both are, in fact, recurrent neural networks. It is important to keep in mind that one (the GRN) controls the development of the other (the phenotype).

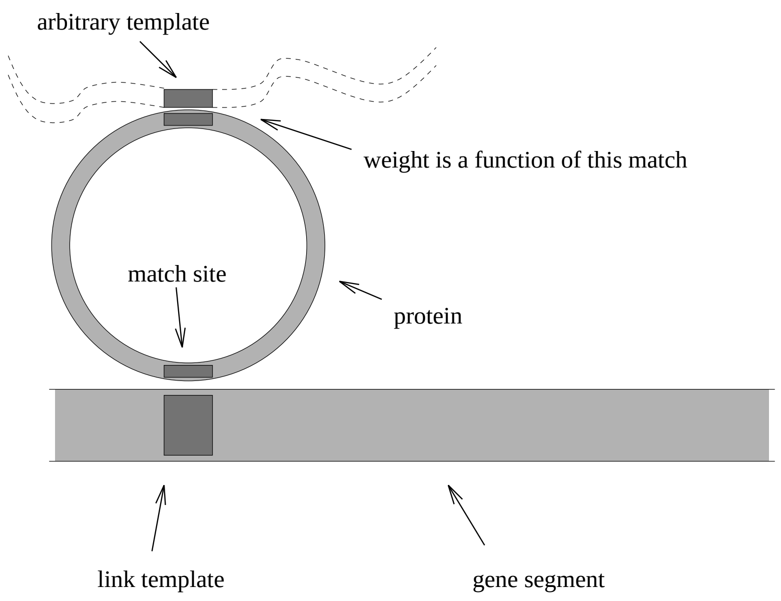

Figure 3: *A protein has been rotated until the match between the link template on the gene and the corresponding protein segment goes over a certain threshold. The contribution this protein makes to the regulation of the gene is then calculated from matching the diametrically opposite side of the protein with an arbitrary fixed template.*

and 64, and then to a decimal between plus and minus one.

In the nucleus of a real cell, which and how many genes any given protein regulates depends on that protein's shape. This shape is arrived at by a complicated folding process, the biological intricacies of which are poorly understood. In the model described in this paper, each protein is a circular string. To work out whether any given protein plays a part in the regulation of any given gene (and hence its product), the protein is rotated over the link template of the gene until, if at all, the value of the match between the template and the protein string segment it is opposite goes over a threshold (see Figure 3). If the threshold is surpassed then the protein plays a role in the regulation of the gene and a link is formed in the GRN between the unit corresponding to the protein and the unit corresponding to the gene's product. The weight on this link is found by matching three characters of the circular protein string, diametrically opposite to where the link template matched, against an arbitrarily defined but fixed template that might conceivably be thought of as part of an RNA Polymerase molecule.

Also at the preprocessing stage, all proteins are assigned to a class depending on the base 4 number expressed by the type regions on their genes. The functionality of a protein within its class is found in similar ways to its regulatory role, as described above (see Section 4.3 for details on each class).

There is one special class of protein that is worth mentioning here as it has a direct influence on the nature of the GRN. These are the signal proteins (see below) that may be emitted from a cell or extra-cellular source, and may enter through the wall of a cell to affect the dynamics of its GRN. A signal protein plays no regulatory role within the cell it is produced, but only within other cells it has entered through the cell wall. The amount of signal protein diffusing *out of* a particular cell is calculated directly from the state of that cell's GRN. However, the activations of those genes (and hence the corresponding units in the GRN) within that cell that are regulated by signal proteins are a function of the amount of signal protein diffusing *into* that cell through the cell wall. See Section 4.4 for an explanation of how the amount of signal protein entering a cell is calculated.

4.3 Protein Classes

There are six classes to which a protein may belong: signals, movers, splitters, differentiators, dendritics and threshold proteins. Each plays a crucial role in the determination of individual cell behaviour and hence to the overall shape and functionality of the final organism. The following is a description of each class's role within the cell.

signals

These proteins diffuse uniformly out from the cell within which they are produced, creating a chemical field (there is no time factor involved) in the space around them. They infuse into cells lying on the resultant chemical gradient wherein they may influence the internal dynamics of the GRN, turning some genes 'on' and others 'off', or interact with mover proteins (see below).

movers

Mover proteins control the way in which individual cells move (see Section 4.4 for details). Each mover protein responds to a subset, particular to that mover protein, of the available signal proteins. This subset is found (during preprocessing) by finding all signal proteins that have a contiguous 3 character string segment that matches a similar 3 character string segment somewhere on the mover protein, with a value that exceeds a certain threshold. This is akin to finding an orientation of two proteins whereby they 'fit together'. If a signal protein matches to a mover protein in this way then the affinity of the mover protein (treated as a weight) for the signal protein is defined as the value returned by matching 3 characters on the signal protein, diametrically opposite to the match site with the mover protein, to a fixed class-specific template.

dendritics

Dendritic proteins control the way in which dendrites grow (see Section 4.5 for details). Each dendritic protein also responds to a subset, particular to that dendritic protein, of the available signal proteins. This subset, and the affinity of a dendritic protein for each of the signal proteins in its subset, is found in exactly the same way as for mover proteins, described above. In addition dendritic proteins also have a threshold associated with them, found by taking the lesser of the maximum match values of two class-specific templates rotated around the protein. This threshold represents the concentration of dendritic protein that must be present in a cell (at differentiation) for the particular dendrite to grow. The dendrite will be excitatory or inhibitory depending on which of the two templates has the lesser maximum match.

splitters

A cell may split in one of two ways, either parallel to the direction of the total force acting on the cell or at right angles to it. Which, if either, of these two options occurs depends on which of two internal variables goes over a certain threshold first. Splitter proteins have two weights between 0 and 1 attributed to them, one for each of these variables, found by taking the maximum match values of two class-specific templates rotated around the protein. At each time step the weighted sum of splitter protein concentrations in the cell is added to each variable; if it splits, both variables are reset.

differentiators

Differentiator proteins work in much the same way as splitter proteins. Each cell contains an internal variable which is *not* reset if the cell splits. At each time step the weighted sum (each differentiator protein also has a weight between 0 and 1 attributed to it, found, in the same way, using a class-specific template) of differentiator protein concentrations is added to this internal variable until it, also, goes over a certain threshold and the cell starts to differentiate.

threshold proteins

After development has ceased, the finished structure is translated into a neural network, with cells becoming units in the network and dendrites becoming the links between these units. Threshold proteins are responsible for setting the thresholds on the units in the final neural network. At differentiation, these thresholds are calculated from the normalized sum of the total concentration of threshold protein within each cell.

4.4 Initial Development

After the genome has been preprocessed, development may begin. A single cell is placed in a two dimensional environment with a number of strategically placed extra-cellular signal protein sources. These protein sources emit chemicals in exactly the same way as a cell does and should not be regarded as somehow having more control over the developmental process than individual cells that emit signal proteins. The concentration of signal protein in the environment drops off with distance from the source according to the expression $(100 - d)/(100 + \sqrt{d})$ where d is the distance. If d is greater than 100 then the concentration is zero. This expression means a concentration of 1 at the signal source dropping to 0 at a distance of 100 (the developmental environment used in Section 3 was a 100 by 100 square) with a non-linearity caused by the $\sqrt{d}$ term which ensures that all signal protein maxima will be located at signal protein emitters (important for dendrite growth, see Section 4.5).

At each time step the concentrations of every signal protein in the local environment of each cell is calculated. This is done by iterating through all signal protein sources and adding the effects of those sources that emit the same chemical. The GRNs of each cell may then be updated synchronously. First, new inputs for all nodes in each GRN are calculated from the weighted sum of recurrent connections and any external signal protein inputs, and then new outputs are calculated using a linear threshold function with lower bound set by the threshold on each node. All internal variables are then updated.

Forces on each cell are calculated by the application of mover proteins to the signal proteins in each cells locality. Each mover protein causes a force proportional to, and in the direction of the maximum gradient of, the weighted sum of the local chemical concentrations of the members of its particular subset of signal proteins. The weights in this sum correspond to the mover protein's affinities for each of the members of its subset. The cell moves in the direction of the resultant of these forces.

If any internal variable (for splitting or for differentiation) has gone over its threshold, then the appropriate action takes place. Note that when a cell splits, each daughter cell is identical to the mother cell in all but a slight positional displacement at right angles to the axis of cleavage.

When the internal differentiation variable within a cell goes over a threshold, the cell differentiates. At this point the cellular concentrations within the cell become fixed, including the levels of signal proteins output from that cell.

4.5 Differentiation and Dendrite Growth

At differentiation the number of dendrites that will grow from a cell is calculated. This is the number of dendritic proteins whose intra-cellular concentration lies above the threshold associated with that protein (see Section 4.3). As with mover proteins, each dendritic responds to its own particular subset of signal proteins. Growth starts from the side of the cell where the weighted sum of the concentrations of each signal protein in this subset is greatest.

Dendrites are guided by a trident shaped 'growth cone' with three sensors, one at the tip of each fork, each responsive to the dendritic protein's subset of signal proteins. At each time step the concentrations at each sensor are calculated, and the position of the base of the trident is updated to that of the trident 'tip' with the highest concentration. In this way the dendrite is steered, at a fixed speed, towards the local maximum of the weighted sum. This will usually be another cell, at which point the dendrite forms a connection with that cell and stops. However, since the sum is weighted, the dendrite may actively steer away from signal protein sources as well as towards them. This creates two sorts of problems. One, dendrites may grow off to infinity, and two, they may go into 'orbit' around local chemical minima. For this reason there is a maximum length to which a dendrite may grow before it is said to be dead.

4.6 Interpreting a Neural Network

Once all cells have differentiated, and all dendrites have either connected or died, the finished structure may be interpreted as an artificial neural network (ANN) which can then be used to control a robot. The architecture and connection matrix of the ANN is taken directly from that of the developed organism. Activation is interpreted to flow in the direction of dendrite growth. Each link is either inhibitory or excitatory (depending on the results of a template match. See Section 4.3). The weight on each link is calculated from the concentrations of the signal protein subset attributed to the dendritic protein responsible. The weighted sum of these concentrations, at the point where the 'growth cone' connects, is divided by the maximum possible value of this weighted sum (i.e. as if all the concentrations were 1.0) to give a number between 0 and 1. This is then scaled appropriately.

Thresholds on the units in the ANN are calculated directly from the concentration levels of threshold proteins in the corresponding cells (see Section 4.3). Input and output units to the ANN correspond to those cells that end up in certain regions of the developmental environment. See Section 3 for an example of how this is done.

5 Issues of Evolvability

There are two properties *essential* to an encoding scheme capable of evolving complex control architectures for robots. Firstly, it must be robust with respect to the genetic operators used by the genetic algorithm (cross-over, mutation, translocation, genome growth etc.), and secondly, it must be capable of 'subroutining', in the sense of encoding for repeated structure in a compact way. The encoding scheme described in this paper displays both of these properties, which are described below, and it is this that makes it potentially very powerful indeed.

5.1 Robustness to the Genetic Operators

If an encoding scheme is to work at all (i.e. to provide the basis for something more than random search) then it must be robust with respect to the genetic operators it is used with. It is crucial, if evolution is to progress, that 'fit' phenotypic traits are not destroyed by the breeding process. Ideally, with the exception of mutation, all genetic operators should cause as little phenotypic disruption as possible. The crossover operator is a special case, since it acts on two genotypes as opposed to one, but I would argue along with [7] that it is only meaningful, in an Evolutionary Robotics context, to use crossover in conjunction with converged populations.

The encoding scheme, reported here, is robust with respect to operators involving translocation and genome growth. This allows two things: firstly, genes that work in tandem (e.g. to create a 'fit' subnetwork in the phenotype) may be relocated next to each other on the genotype, thus minimising the chance of their separation by crossover, and secondly, open ended evolution becomes possible, where phenotypic complexity is not restricted by the original size of the genome. This robustness follows from the simple reason that, using template matching, parts of the genotype address other parts of the genotype (to form the links of the GRN) by the character sequences that occur there, and not, as in many encoding schemes, by a function of the region's location on the genome. If these sequences of characters are relocated on the genotype, or if the genotype is allowed to grow or shrink, it will have minimal effects on the shape and nature of the GRN.

Ideally, mutating any genotype should, *in the main*, provide only minor changes to the phenotype (of a degree proportional, in some sense, to the changes made to the genotype). However, especially in the case of converged population genetic algorithms, small mutations to the genotype should occasionally result in large changes to the phenotype [3]. Because of the way a developmental scheme works,

[3]Since we are, in effect, in charge of the physics under which development occurs, it is also possible to restrict the form that large scale changes may take. This may well be desirable since

small scale changes to the GRN that controls development may nevertheless produce large scale changes in the phenotype (whole new portions of network may grow, or sub-structures may get repeated, see below). Also a small change to the GRN may do nothing but slightly alter one parameter of the phenotype. Slight mutations to the GRN, then, provide the right range of changes in the phenotype.

What is less obvious is why slight changes to the genotype, in the encoding scheme reported here, should result in slight changes to the GRN. Section 4.2 explains how the GRN is decoded from the genome using a variety of template matching operations. The problem is that if two or more templates overlap then a single mutation at this point may result in a disproportionately large change to the GRN. In order to minimize this possibility, the template matching routines were designed to extract the maximum amount of information from templates that were as short as possible. Also, the ratio of protein-length to template-length (e.g. the link templates on the gene and the class-specific templates used in preprocessing) was made as large as possible. Inevitably, though, a certain amount of overlap will occur. However, in practice, evolution very quickly favours those genotypes that display low amounts of overlap, since mutation of these genomes is far less disruptive in general and fit individuals are more likely to have fit offspring. The epistasis caused by this overlap is then almost entirely 'bred out' of the population.

5.2 Encoding Repeated Structure

The positions of the extra-cellular signal protein sources within the developmental environment restricts the final neural network architectures that are possible. For example, in the experiment detailed in Section 3, development unfolds in conjunction with only a single extra-cellular signal protein source, ensuring bisymmetry in the resultant neural networks. It must be stressed that, with the addition of more extra-cellular protein sources (in asymmetric positions), fully developed networks would not *necessarily* be asymmetric, since the signal protein produced by any given source may well not template match anywhere on the genome (see Section 4.2), and thus would not affect the dynamics of any cell within its chemical field.

Each cell, during the developmental process, will only respond to a subset of the signal proteins within which it is immersed. Different cells at different times will respond to different subsets, depending on the particular positions of their GRNs in dynamical state space. This allows the developmental process to buy in or out of symmetry, and the same applies to buying in or out of repeated structure. For example, imagine two cells on opposite sides of a developmental environment which is asymmetric with respect to signal protein concentrations. Now even though each cell is immersed in a different set of signal proteins, it so happens that they both respond to the same subset, and that the environment is symmetrical with respect to this subset. If both of the cells' GRNs are at the same position in dynamical state space at this time, then they will display the same behaviour, since, in effect, their internal and external influences are identical, and they may split, move and differentiate to produce identical subnetworks on either side of the developmen-

some forms of large scale change, such as the insertion of a totally random subnetwork in an ANN, are less likely to be useful than, say, the repetition of an already existing subnetwork in the same ANN. In [5], Goodwin argues that something like this sort of restriction plays a major role in evolution.

tal environment. However, cells elsewhere in the developmental environment may respond to different signal proteins subsets and may, therefore, develop asymmetrically. The final developed network will be asymmetric as a whole but with two identical subnetworks.

6 Conclusions

This paper represents an attempt to harness the developmental power of morphogenesis to the still young discipline of Evolutionary Robotics. A biologically inspired encoding scheme has been outlined, capable of encoding repeated structure and symmetry in a compact way, which is robust to the wiliest of genetic operators. It has been successfully used to evolve two simple behaviours for a small mobile robot.

It is the author's hope that with time, a descendant of the current encoding scheme will make possible the evolution of far more complicated behaviours than those outlined in Section 3, and maybe even robots themselves (see [2]). The scheme's ability to cope with open ended, incremental evolution [7], while building up a 'library' of useful subnetworks on the genome, may finally provide the evolutionary robotocist with the means to compete (in terms of time and resources) with other more traditional methods of control architecture design, a necessary step if evolutionary robotics is to fulfill its early promise.

Acknowledgements

Nick Jakobi is supported by a postgraduate bursary from the School of Cognitive and Computer Science at the University of Sussex. Many thanks to Inman Harvey for allowing me, and encouraging me, to build on the proposed encoding scheme outlined in [8].

References

[1] R. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–59, 1991.

[2] Frank Dellaert and Randall D. Beer. Co-evolving body and brain in autonomous agents using a developmental model. *Technical Report*, (CES-94-16), 1994.

[3] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural -network driven robot. In D. Cliff, P. Husbands, J.A. Meyer, and S. Wilson, editors, *From Animals to Animats*, volume 3, 1994.

[4] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, Massachusetts, USA, 1989.

[5] B. C. Goodwin. The evolution of generic form. In J. Maynard Smith and G. Vida, editors, *Organizational constraints on the dynamics of evolution*, Proceedings in Nonlinear Science, pages 107–117. Manchester University Press, 1990.

[6] C.F. Graham and P.F Wareing. *Developmental Control in Animals and Plants.* Blackwell, Oxford, 2 edition, 1984.

[7] I. Harvey. Species adaptation genetic algorithms: the basis for a continuing saga. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354, Cambridge, Massachusetts, 1992. M.I.T. Press / Bradford Books.

[8] I. Harvey. *The Artificial Evolution of Adaptive Behaviour.* PhD thesis, University of Sussex, 1993.

[9] I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, 1994.

[10] P. Husbands and I. Harvey. Evolution versus design: Controlling autonomous robots. In *Integrating Perception, Planning and Action: Proceedings of the Third Annual Conference on Artificial Intelligence, Simulation and Planning*, pages 139–146. I.E.E.E. Press, 1992.

[11] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In F. Moran, A. Moreno, J.J. Merelo, and P. Chacon, editors, *Advances in Artificial Life: Proc. 3rd European Conference on Artificial Life*, pages 704–720. Springer-Verlag, Lecture Notes in Artificial Intelligence 929, 1995.

[12] K-Team. Khepera users manual. EPFL,Lausanne, June 1993.

[13] Stuart A. Kauffman. *The Origins of Order.* Oxford University Press, 1993.

[14] S. Nolfi, O. Miglino, and D. Parisi. Phenotypic plasticity in evolving neural networks. In *Proceedings of the PerAc'94 Conference.* IEEE Computer Society Press, 1994.

[15] M. Ptashne. *A Genetic Switch.* Cell Press, 1986.

[16] J. Vaario. From evolutionary computation to computational evolution. *Informatica*, 1994.