

Evolution of repressilators using a biologically-motivated model of gene expression

Barry Drennan¹ and Randall D. Beer^{1,2}

¹Department of Electrical Engineering and Computer Science

²Department of Biology

Case Western Reserve University

btd@po.cwru.edu, rxb9@po.cwru.edu

Abstract

The evolution of development is an important ongoing question in developmental biology, and to study it, we examine evolution in the context of fundamental processes underlying development, gene expression and regulation. In this paper, we present the results of utilizing a biologically-motivated model which implements gene expression and regulation to evolve genetic regulatory networks that express proteins in an oscillatory manner similar to a "repressilator" network. The networks were artificially evolved via genetic algorithm by manually bootstrapping the initial population with a set of randomly generated networks. The genetic algorithm then utilizes a two-part strategy combining sudden small changes which alter connections in the network with a gradual "paring down" of the network to expose higher-fitness components.

Introduction

How did development evolve? This is one of the more difficult questions faced by developmental biologists. As Wray notes (2003), studying the evolution of development in a purely biological context "is a daunting task." The size of the genome compared to the size of each mutation, the likelihood of a mutation resulting in no phenotypic change, and the interference of environmental effects in the developmental process all make studying the evolutionary track of development vastly more difficult. While there are techniques for understanding the genes that contribute to development (Medina 2005), and portions of the evolutionary history of development can be reconstructed, the field of evolutionary developmental biology would benefit from new techniques unencumbered by the difficulties of studying biological organisms.

The artificial life community has its own take on the topic of the evolution of development. Several works have used the concept of indirect genotype to phenotype mappings to devise schemes for the evolution of morphology or neural networks through a developmental process (Stanley and Miikkulainen 2003). Kitano (1990) and Gruau and Whitley (1993) evolved artificial neural networks under a developmental scheme based on grammar trees and inspired by biological cell division. Sims (1994) used a directed graph representation to evolve the morphologies of co-evolving agents. Cangelosi *et al.*

(1994) devised a model of neural development based on cell division and migration as well as axon growth and branching, using a binary genome to describe developmental rules. However, as Dellaert and Beer (1994) note, these more abstract models are not motivated by the possibility of using them to study biological phenomena and therefore lack the biological realism necessary for that application.

More recently, researchers have taken up more biologically-motivated methods and applied them to studying model biological systems. Bongard and Pfeifer (2003) have used a developmental process based on genetic regulatory networks to evolve artificial agents. Geraud and Wiles (2003) have taken this regulatory network approach and applied it to studying early cell division in *C. elegans*, though they do not use it in the context of evolution.

The choice of mechanism for representing the genotype can have a tremendous effect upon the ease of evolving high-fitness individuals. As Dellaert and Beer (1996) discovered, an overly flexible genome representation can inhibit a genetic algorithm from discovering solutions compared to a more abstract representation specifically designed for the problem, yet the high flexibility of DNA is exactly what biological evolution must contend with.

In order to examine the question of evolution of development, our work models the problem in terms of basic biology, with abstractions included only to reduce computational complexity and to focus on the questions at hand. We focus on the mechanisms underlying the developmental process, such as gene regulation. We have developed a biologically-motivated model (Drennan and Beer 2004) for studying the evolution of development by studying gene regulation, and model the genome as a string of DNA-like bases which are transcribed and translated into proteins that regulate gene expression.

This paper presents the results of using this model to evolve genomes capable of mimicking the behavior of a network called a repressilator (Elowitz and Leibler 2000). Specifically, we demonstrate that, although evolution fails when initialized with completely random genomes, repressilators do evolve reliably when the initial population is seeded with genomes containing randomly-generated regulatory networks. In addition, examination of successful evolutionary runs reveals a strategy where the processes of construction and pruning are both essential.

Methodology

Model: Genomes and Proteins

In biology, DNA encodes genetic information, consisting of four distinct bases, or “characters”: adenine, guanine, cytosine, and thymine (A, G, C, and T). Bases are arranged in strands of DNA to form substrings called genes, which are interpreted by the cell’s chemical machinery to produce proteins. Our model also uses genetic strings consisting of four characters, but to reinforce the differences between biology and our model, we name our bases **w**, **x**, **y**, and **z**.

In biological gene expression, DNA is used as a blueprint for transcription of messenger RNA (mRNA). This is then translated by ribosomes into sequences of amino acids called proteins. The mRNA is translated in three-base segments called codons, each indicating that a particular amino acid should appear next in the protein.

The transcription of genes can be regulated through the binding of certain proteins onto segments of DNA. In biological systems, the interactions of these proteins with three regions of DNA – promoter, enhancer, and repressor, usually located upstream from the corresponding genes – govern when a gene can be transcribed.

Promoter regions indicate the start of genes. These regions usually include a short segment of DNA consisting of thymine and adenine in one of a number of sequences (TATAAT, for example), which allow transcription factors to bind to and separate the two strands of DNA.

Upstream from the promoter region is often found another region of DNA which can be bound to by proteins which assist the transcription machinery in binding to the DNA. These regions thus enhance the transcription of the associated gene. Downstream from the promoter region may be a segment of DNA which can also be bound to by proteins. When these regions are occupied, the presence of a binding protein blocks the progress of the transcription machinery, preventing the gene from being expressed. These two effects – enhancement and repression – form the core mechanism by which gene expression is regulated.

We model transcription similarly in most respects to the biological equivalent. Each gene consists of the three regulatory regions – promoter, enhancer, and repressor – plus the coding region (Figure 1). While genes in biological systems are located via the promoter region, each gene in our model is located by the existence of a start codon (either **www** or **wxx**), and each start codon in the genome is evaluated in sequence to determine whether its

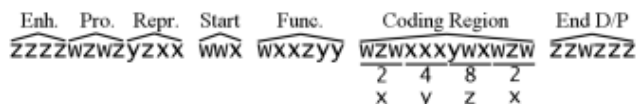


Figure 1. A model gene. Labeled are the different regions of the gene: enhancer, promoter, repressor, start codon, domain function, coding region, and end-of-domain/protein codons. Below the coding region are the amino acid sequence and the corresponding gene sequence it binds to.

gene is expressed and how much protein is produced. From there, the promoter region is identified as the sequence most closely matching the ideal promoter sequence (**wzwz**) within 10 upstream bases of the start codon, and the closeness with which that matching occurs indicates the basal transcription rate of the gene.

The promoter region also indicates the locations of the enhancer and repressor regions – the enhancer is immediately upstream from the start of the promoter, while the repressor occurs immediately downstream from the end of the promoter. The sequences of proteins are evaluated to determine whether they bind to the enhancer or repressor regions. The rate of gene expression is increased if the enhancer region is bound. However, if the repressor region is bound, the expression of the gene is blocked altogether.

Each codon represents an amino acid, the building block of the protein. In our model, we implement 16 different amino acids. The functions of these amino acids are determined in part by their locations within the protein. One particular amino acid occurs at the start of each protein, and another occurs to signify the end of each domain. A special codon denotes the end of the protein.

Our model abstracts the phenomenon of protein folding into a simple process of interpreting consecutive regions of amino acids as domains. Each domain in our model consists of three parts: a function designator, an optional data string, and an end-of-domain amino acid. Domain functions are determined by the first two amino acids within the domain. In this work, all proteins generally have the “gene bind” function, where the data string indicates which genetic bases can be bound to by the protein.

Repressilator

We have chosen as a basis for our evolution task the sorts of oscillations produced by the repressilator (Elowitz and Leibler 2000). The repressilator is a regulatory network consisting of three genes, each of which produces a protein which inhibits the expression of the next gene, forming a cycle (Figure 2). As the concentration of one of the three produced proteins increases, the next gene in the cycle is inhibited, and the concentration of the second protein falls. That permits expression of the third gene, and as the third protein increases in concentration, the first gene becomes inhibited, allowing expression of the second gene, and so forth.

This circuit has been implemented in *E. coli* by Elowitz and Leibler, and forms a stable base case that is grounded in biology. Because of this, and due to the simplicity of the designed repressilator network, we use the behavior exhibited by the repressilator as the basis for fitness evaluation in the genetic algorithm, as described below.

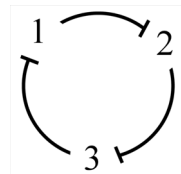


Figure 2. The three cyclically-inhibitive genes of the repressilator.

	Repressors	Function	DNA binding sequence	
WZWZ	xxyy	wwx	wxxzyy	xxx yww wzw wzw
WZWZ	yzxx	wwx	wxxzyy	wzw xxx yww wzw
WZWZ	xyzx	wwx	wxxzyy	wzw wzw xxx xxx

Figure 3. The designed repressilator genome.

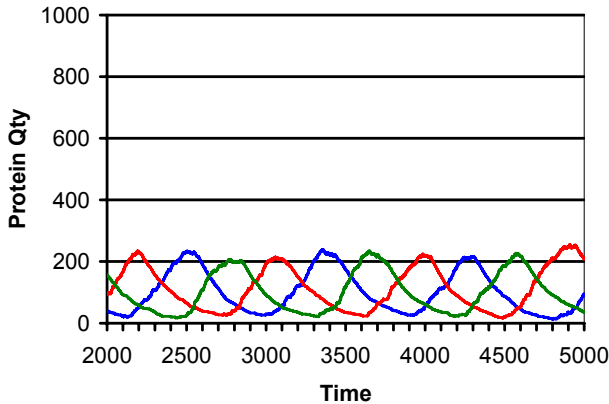


Figure 4. Protein output from the designed repressilator.

We have implemented the repressilator in our model with the genome shown in Figure 3, which produces the protein concentrations shown in Figure 4.

Genetic Algorithm

Our model includes a facility for evolving genomes via genetic algorithm (Holland 1975). The GA includes five genetic operators, based on genetic mutations seen in biology (Kimball 2005), as well as elitist selection, which, though not particularly biologically realistic, was included since high mutation rates can otherwise cause high-fitness individuals to fall out of successive generations.

Single-base mutation randomly selects a base in the genome and replaces it with one of the four bases. Block insertion constructs a random genetic string and inserts it at some point in the genome. Block deletion randomly selects a substring of the genome and removes it altogether. Block copy selects a substring of the genome and duplicates it, inserting the duplicated substring at another point in the genome. Crossover selects a substring from a second individual and inserts it into the first genome. Individuals can be selected into each new generation unchanged, either randomly or as part of elitist selection.

We wished to evolve genomes that would accomplish the same task as the repressilator, though not necessarily using the same strategy for obtaining oscillations that the designed repressilator uses. We therefore developed a fitness function that would reward out-of-phase oscillation in three or more protein concentrations:

0. Start with zero fitness, an empty exclusion list of length 2, and protein concentrations of zero.
1. Skip transient period (2000 time units).
2. Find a protein not on the exclusion list whose concentration is above a threshold (150 proteins).

3. Track that protein over time until it peaks and then reduces in concentration to 75% of its peak value.
4. Push that protein onto the exclusion list, and remove the protein that falls off the other end of the list.
5. Increase the fitness score by 1.
6. Go to step 2, unless the evaluation time has elapsed (total 12000 time units).

Using this fitness measure, the designed repressilator achieves a repeatably-obtainable fitness of 34. The highest fitness ever seen from an individual throughout testing (more than 1.5 million) is 35. All such individuals exhibit oscillation of protein concentrations, so we are confident that this fitness measure is suitable for describing the desired behavior. Note that each individual – including individuals that were not mutated from the previous generation – is evaluated each generation. Due to stochasticity in the model, fitness can vary by a fairly wide margin (as much as 20% in some cases).

Results

Random Genomes and Random Regulatory Genes

Our first experiments involved purely random seeding – that is, generating a population where each genome consisted of a completely randomly-generated string. Unfortunately, the necessary features of a genome that can produce proteins are stringent enough that we were unable to evolve genomes that produced proteins at all. The fitness of a genome that does not produce proteins is zero, so the GA's performance amounts to a random search.

We next tried evolving based on seed individuals consisting of protein-generating genes, where the basic structure for a gene was present. We randomly generated the repressor regions of each gene as well as the portion of the coding region which corresponded to the binding sequence of the produced proteins. That is, the regulatory portions of each gene did not necessarily match the corresponding portion of any other gene. Thus, while we had genomes that could produce proteins, these genomes were not capable of forming the networks necessary to regulate oscillating protein concentrations. These failures were due to the unlikelihood of randomly arriving at the proper combinations of genes, given that proteins that do not regulate each others' production score zero fitness.

Random Regulatory Networks

We dealt with this issue of a large, flat fitness landscape by seeding the initial population with randomly-generated genetic regulatory networks. This provided genes that produced proteins which regulated expression of other genes present in the network, thus starting the search in a nonzero region of the fitness landscape.

Note that, given the string-matching mechanism by which gene expression is regulated in our model, it is not possible to construct an arbitrary network based on a graph where one gene corresponds to one node of the graph.

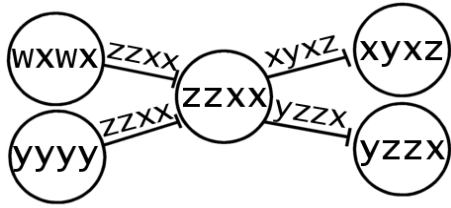


Figure 5. Shown is an example regulatory network. Each node represents a repressor sequence, and each edge represents a gene. The node in the center is regulated by two genes, and its outgoing edges indicate two different genes that have *zzxx* as their repressor sequence.

However, arbitrary networks can be constructed where graph *edges* correspond to genes, and nodes correspond to the repressor strings that occur before the genes (Figure 5).

We generate random networks by first generating a connection matrix describing the network graph. The matrix is populated with 1's and 0's depending on a connectivity parameter which indicates the probability of each possible edge being present in the completed graph (in all of our experiments, we set this at 50%). This matrix is then used to construct a genome implementing the network. Each "1" entry in the matrix indicates one gene that should be present in the genome. The row indicates the repressor sequence that should appear before the gene, and the column indicates the repressor sequence that the coded protein should bind to.

Our initial experiments with randomly-generated networks proved successful, with three GA runs producing networks with fitness of 30 or higher. In these cases, inspection of the networks over evolutionary time revealed that nearly all of the seed individuals already contained a standard repressor, and the GA gradually revealed that repressor over time by mutating other genes to prevent them from participating in the regulatory network. About 99.86% of randomly-generated networks contain repressors somewhere within the network, although in most cases, the rest of the connections within the network prevent the repressor from reaching its full potential.

To require that the GA actively create a network that produces oscillatory behavior, rather than simply uncovering a pre-existing subnetwork, we began including only seed individuals that do not contain a repressor. Our experiments show that the GA is capable of taking the step of forming repressor-style networks from random networks where no repressor existed previously.

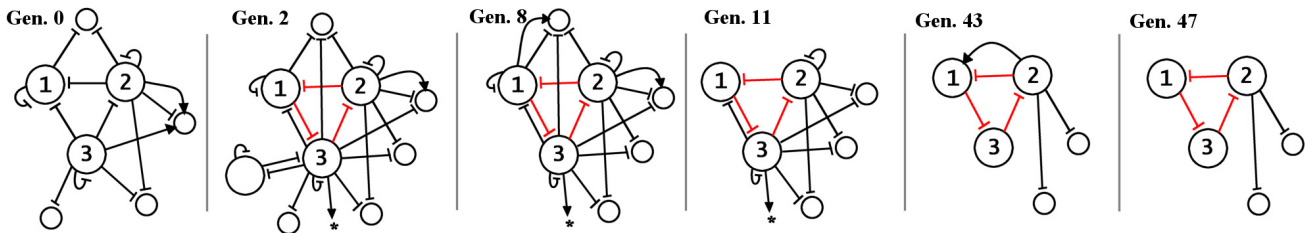


Figure 7. The evolution history of the best individual in one experiment. The repressor subnetwork formed at generation 2 is shown in red. Inhibitory connections end in bars, and excitatory connections end in arrowheads. Small circles indicate nonregulatory genes. The asterisk indicates that the node has excitatory connections to every other node in the network.

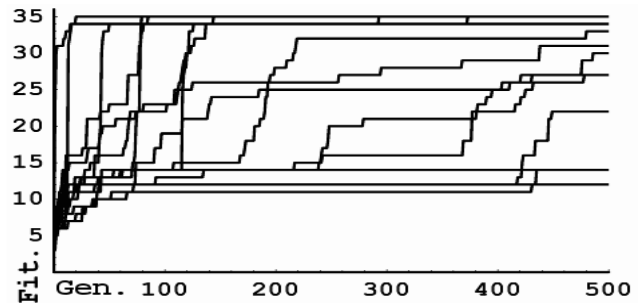


Figure 6. Fitnesses over time for the 16 GA experiments. The first 500 generations are shown.

The rest of the experiments described herein use randomly-generated regulatory networks where the initial seed population is required not to have any repressors contained within the networks. Each generated network consisted of 6 nodes with roughly 50% connectivity.

We conducted 16 experiments with this setup. Of these experiments, after 1000 generations, 3 failed altogether (produced a maximum fitness score below 20), 1 failed marginally (produced a maximum fitness from 20-29), 5 succeeded marginally (produced a maximum fitness of at least 30 but below the standard repressor's fitness of 34), and 6 succeeded fully (produced a maximum fitness of at least 34). The maximum fitness seen in any experiment was 35. The fitnesses over time are shown in Figure 6.

Evolution History and Lineages

Out of the 16 experiments completed, we have chosen one as an example for analysis in this paper which shows the primary ways in which the GA develops a network capable of generating oscillating protein concentrations. The example case reached a fitness of 34 within 50 generations. We therefore trace back the lineage of the best individual from generation 50 to determine how it evolved (Figure 7).

The GA accomplished this by first forming a repressor at generation 2, through a crossover with a short segment of another individual. This deleted five genes from the network and replaced the repressor sequence of one gene with a sequence from the other genome that did not appear in the first parent at all. The inclusion of this new repressor sequence created a new regulatory connection, completing a repressor-like cycle. At this point, however, the genome received a fitness of 0, because the three genes participating in the

repressilator-like cycle were encumbered by self-repressing links, preventing their expression.

While the first repressilator in the lineage of generation 50's best individual appeared in generation 2, its presence in the genome was maintained because of the overall low fitness of the population, during which time a crossover at generation 8 caused the loss of some genes from the end of the genome but produced no increase in fitness. It then experienced another crossover in generation 11. This change removed one of the self-repressing links and allowed the network to achieve a fitness of 12.

At generation 43, another crossover caused the loss of more genes, including the gene acquired from the crossover in generation 2. During the interim, other genes were modified such that the deletion of genes at generation 43 did not destroy the existing repressilator. This individual achieved a fitness of 30. During the remaining generations, the GA performed one more crossover which eliminated an excitatory link, producing a repressilator driving two nonregulatory genes.

Across the successful GA runs, we observed a common two-part theme. One, the GA tended to alter or create links in networks suddenly, through mutations in the repressors of certain genes. A very few of these mutations were required throughout each run to form a repressilator as a subnetwork. Two, the GA pruned the network throughout the run, gradually removing unnecessary connections and exposing any repressilators that may have formed earlier.

The excitatory connections that occasionally occurred as a byproduct of the initial network formation were often used during early evolution to complete a repressilator where only two out of three inhibitory links existed, but these were eventually supplanted by inhibitory connections and removed. An excitatory connection can participate in a repressilator, but this is suboptimal; such genomes typically receive a fitness in the low 20s.

Discussion

Network Formation

Our early experiments indicated that evolution from scratch (i.e., a purely random genome, or a set of regulatory genes that are not necessarily interactive) is prohibitively difficult, a finding consistent with Dellaert and Beer (1996). This difficulty results from the large number of bases that must appear in a particular order to cause protein expression. Several bases must match a set of constraints, and these constraints are increased if a particular protein function is desired.

However, biological evolution does not occur in a vacuum. While we have subsumed factors such as the metabolic processes of our model into the hardcoded workings of our software, these factors in biological organisms instead occur within the genome itself alongside the genes that may participate in new functions through evolution. In significantly evolved organisms, these genes

form a pool of genetic material available to copy and modify, and in many cases, these genes already interact in a regulatory fashion. When we seed the GA with genomes containing random networks, we provide such a pool of genetic material to the organism, and the GA modifies these genes to produce a repressilator-like network.

We believe that the inclusion of at least an arbitrary genetic network as a seed for the GA is crucial to the successful evolution of specific networks. We also believe that the general concept of bootstrapping evolution by providing basic precursor material holds true for artificial evolution in any situation where the agent's genome is highly flexible; artificial life research may be able to use more biologically-realistic genome representations by using such a technique. The concept may also hold true in biological systems, where there is already great flexibility in what DNA-based genomes can specify.

“No Repressilator” restriction

During our initial evaluations of the three unconstrained GA runs (see “Random Regulatory Networks” above), we noted that practically all of our initial populations contained repressilator subnetworks, and the GA's strategy simply consisted of breaking enough genes to uncover the subnetwork. We therefore expected that by preventing repressilators from appearing in the initial population, the GA searches would experience far more difficulty in achieving high fitness. However, the formation of a new repressilator within a random regulatory network was trivial. In most cases, the strategy exhibited by the GA involved constructing a repressilator by modifying the repressor sequences preceding each gene, either by copying the repressor sequence from another gene or by mutating a single base of an existing repressor sequence.

In some of our test cases, the GA made use of the fact that several genes may initially have the same repressor sequence. In prokaryotes, many genes are contained within “operons”, which are sets of genes governed by the same “operator” – meaning that the same regulatory factors can enhance or repress many genes simultaneously. Subsequently being able to copy a single gene out from an operon during evolution may provide a useful strategy for forming certain kinds of regulatory networks.

Network Reduction

The other major strategy utilized by the GA was the gradual paring down of arbitrary networks to reveal subnetworks with higher fitness. The GA would make use of crossover and block deletion to remove all or part of genes that interfered with a higher-fitness subnetwork, and would use block insertion and block copy to introduce errors into genes to cause them no longer to become functional. In some cases, the GA would maintain certain subnetworks, such as excitatory connections that existed where an inhibitory connection was not present, and then relatively quickly eliminate those connections once the proper inhibitory connections were in place.

Even when a network had already achieved a fitness of 34, permitting the GA to continue running often resulted in the removal of extra nonregulatory genes that had no effect on fitness. Admittedly, it is difficult to quantify the occurrence and size of these genetic operations in biological evolution, and is even more difficult to arrive at equivalent values in our more abstract system, but the presence of copious amounts of “junk DNA” in biological systems may indicate many genes that were at one point functional but were later disposed of by evolution.

Evolutionary Strategy?

The combination of the “network reduction” strategy with the need for a sizable initial base of genetic material for the GA to work with suggests a possible strategy for biological evolution: the building up of large sets of genes, possibly through gene duplication, which interact in arbitrary, but not detrimental, ways, followed by random changes made to these genes and their interactions combined with a period of culling in which the genes which do not contribute positively to the species are gradually eliminated. This hypothesis is mirrored by Valentine (2000) in the context of metazoan evolution.

The theme of building up large networks, modifying them, and culling the less useful portions occurs elsewhere in biology, though on a far shorter time scale. For example, neural development in many animals involves producing large numbers of neurons and synapses which are then culled based on which synapses are used more (van Ooyen 1994, Henderson 1996). We believe that this theme as a possible evolutionary strategy warrants further research in the realms of both biology and simulation.

Acknowledgments

This work was made possible in part by the NSF IGERT program in Neuromechanics at Case Western Reserve University and by NSF grant EIA-0130773. We also would like to thank Eldan Goldenberg, Sean Psujek, and Chad Seys for their tremendously helpful insight.

References

Bongard, J.C. and Pfeifer, R. (2003) Evolving Complete Agents using Artificial Ontogeny. In *Morpho-Functional Machines: The New Species (Designing Embodied Intelligence)* 237-258. Springer-Verlag, Berlin.

Cangelosi, A., Parisi, D., and Nolfi, S. (1994) Cell division and migration in a ‘genotype’ for neural networks. *Network: Computational Neural Systems* 5:497-515.

Dellaert, F. and Beer, R.D. (1996) A developmental model for the evolution of complete autonomous agents. In *From Animals to Animats 4: Proc. of the Fourth Int’l Conference on Simulation of Adaptive Behavior* 393-401. MIT Press.

Dellaert, F. and Beer, R.D. (1994) Toward an evolvable model of development for autonomous agent synthesis. In *Proceedings of Artificial Life IV* 246-257. MIT Press.

Drennan, B. and Beer, R.D. (2004) A model for exploring genetic control of artificial amoebae. In *Artificial Life IX: Proc. of the Ninth Int’l Conference on the Simulation and Synthesis of Living Systems*, 381-386. MIT Press.

Elowitz, M.B. and Leibler, S.A. (2000) Synthetic gene oscillatory network of transcriptional regulators. *Nature* 403:335-338.

Gear, N. and Wiles, J. (2003) A Gene Regulatory Network for Cell Differentiation in *Caenorhabditis elegans*. In *First Australian Conf. on Artificial Life, ACAL’03*.

Gruau, F. and Whitley, D. (1993) Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect. *Evolutionary Computation* 1(3):213-233.

Henderson, C.E. (1996) Programmed Cell Death in the Developing Nervous System. *Neuron* 17(4):579-585.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Kimball, J.W. (2005) Kimball’s Biology Pages. <http://biology-pages.info/>

Kitano, H. (1990) Designing neural networks using genetic algorithm with graph generation system. *Complex Systems* 4:461-476.

Medina, M. (2005) Genomes, phylogeny, and evolutionary systems biology. *Proc. of the Nat’l Academy of Sciences* 102(Supp 1):6630-6635.

Sims, K. (1994) Evolving 3D morphology and behavior by competition. *Proceedings of the Fourth Conference on Artificial Life* 28-39. MIT Press.

Stanley, K.O. and Miikkulainen, R. (2003) A Taxonomy for Artificial Embryogeny. *Artificial Life* 9(2):93-130.

Valentine, J.W. (2000) Two genomic paths to the evolution of complexity in bodyplans. *Paleobiology* 26(3):513-519.

van Ooyen, A. (1994) Activity-dependent neural network development. *Network: Computation in Neural Systems* 5:401-423.

Wray, G.A. (2003) Transcriptional regulation and the evolution of development. *Int. J. of Developmental Biology* 47:675-684.