The Artificial Evolution of Real Intelligence by Natural Selection

Alastair Channon and Bob Damper
Image, Speech & Intelligent Systems Research Group
University of Southampton, Southampton, SO17 1BJ, UK
adc96r@soton.ac.uk http://www.soton.ac.uk/~adc96r

Abstract

This paper outlines a preliminary step towards the long-term goal of intelligent artificial life. Evolutionary emergence via natural selection is proposed as the way forward, in combination with other biologically-inspired principles including the developmental modularity of neural networks. In order to develop and test the ideas, an artificial world containing autonomous organisms has been created. Its underlying theory and construction are described. Resulting emergent strategies are reported both from an observer's perspective and in terms of their neural mechanisms. The results prove that the proposed approach is viable and show it to be an exciting area for further research.

1 Introduction

The interest of this work is the generation of intelligence worthy of comparison with that found in natural life, even if only at a rudimentary level. This requires the creation of systems that adapt to behave in 'intelligent' ways within an environment, without being given any information about how to do so.

The approach taken here is based on the only known systems exhibiting such intelligence: organisms with nervous systems as evolved in nature. Thus the method used is evolution by *natural* selection, and the objects being evolved are organisms controlled by artificial neural networks. Other related principles believed necessary are also copied from nature.

The use of natural selection as opposed to the fitness-function based artificial selection of conventional genetic algorithms is an important aspect of this work. Increasingly complex behaviours can emerge, as in nature, via the co-evolution of similarly-capable systems.

One issue worth clarifying at the start is that of functional validity with respect to the destination environment, or 'situation within a world'. Brooks [3, 4] puts forward the argument that incremental development (including evolution) must take place within the world that the objects are intended to operate in. This is to avoid problems common in traditional artificial intelligence, often caused by a divide between a system and the real world. So, for example, some people argue that robots that are to operate in the real world must be evolved (or at least evaluated) in the real world. However, the organisms considered here only ever operate in an artificial world and so there should be no concern about them being evolved in that world. Their world is not a simulation and so suffers none of the problems that occur when trying to use a simulation to evolve robots for the real world. Where the artificial world differs from our world (however greatly), there is no problematic error. There is simply a difference.

The rest of this paper is organised as follows:
- Background
  - Evolutionary emergence
  - Development in artificial neural networks
- Underlying theory
- The artificial world
- Results
- Conclusions

2 Background

2.1 Evolutionary Emergence

Emergence is related to novel, unexpected behaviours. This relationship often leads to definitions of emergence that are dependent on the predictive ability of an observer and demand a once-only instance of emergence. According to such definitions, an exact repetition of a process that yielded an emergent property does not yield an *emergent* property, since the observer is no longer surprised.

Steels [21] gives temperature and pressure as examples of emergent properties that would not be classified as emergent by such a definition. He uses emergence to refer to ongoing processes which produce results invoking vocabulary not previously involved in the description of the system's inner components ('new descriptive categories' [21]). This is the definition used in this paper. Evolutionary emergence is simply emergence resulting from an evolutionary process.

2.1.1 An Example: 'Farmers and Nomads'

Sannier and Goodman [20] used a genetic algorithm to evolve genomes within a two-dimensional toroidal artificial world containing 'food'. Each individual (genome) was given the ability to detect the conditions in its internal and immediate external environments. An individual's 'strength', which is deducted from its parents' at birth, increases on consumption of food and decreases in each time step (and upon reproduction). To reproduce, an individual's strength must be above a threshold. An individual dies if its strength drops below a lower threshold.

The genomes encode rules which allow them to move in one of eight directions (N,NE,E,SE,...) with program branching conditional on the presence of food in the eight neighbouring locations.

In the experiment reported, food was restricted to two 'farm' areas, spaced apart in the toroidal world. The level of food in a farm varied periodically; when one farm was having its 'summer' the other would be having its 'winter'. A farm's potential was set lower the more it was either over-consumed or neglected (under-consumed) during the previous period.

Two classes of individual emerged from the evolution: 'farmers' who stayed put in one of the farms, their populations rising and falling with the 'seasons', and 'nomads' who circled the world, moving in such a way that they passed through both farms during their summers. The nomad population would increase as it went through a farm and decrease as it moved through the area without food. Notice the new descriptive categories 'farmer' and 'nomad'.

Groups of individuals from each category were extracted from the total population and tested in the absence of the other category. Whilst farmers could survive without nomads, it was found that nomads needed farmers so that the farms would not be neglected between visits.

The (relevant) important feature is the emergence of the two classes of individual. Never was it specified that they should arise. Evolution produced them simply because they perform better than other solutions within the environment. The only information given, above the genetic algorithm and external environment, was the available actions (moves) and conditionals (food in neighbourhood?). It could be said that the system outperformed the specifications given it.

2.1.2 SAGA

Most conventional genetic algorithms set out to solve a well-defined optimisation problem. For such problems the individuals (solutions) are generally encoded on a fixed-length genotype, often directly (using a bijection between the solution set and the chromosome set). But we wish to go beyond using evolutionary algorithms to solve *specific* problems, towards using them for emergent systems that perform well in general. Genetic algorithms with fixed-length genotypes cannot be used to evolve increasingly impressive systems as natural evolution has done, so we need to use variable-length genotypes.

The subject of how genotype lengths should change has been addressed by Harvey's 'Species Adaptation Genetic Algorithm' (SAGA) theory [9, 10]. He argues (and demonstrates) that the changes in genotype length should take place much more slowly than crossover's fast mixing of chromosomes. The population should be nearly-converged, evolving as species; mutation rates should be low enough that they do not disperse the species (in genotype space) or hinder the assimilation, by crossover, of beneficial mutations into the species. The idea of species is not engineered in, but rather a result of this theory. A new species arises (emerges) when a progenitorial species splits into separate ones. A species becomes extinct once all its members have died.

In slightly more detail, taking into account gene duplication and genotype to phenotype (ontogenesis) systems, Harvey is arguing that the *complexity* of the information contained within the genomes of a species should change slowly, relative to the assimilation of advantageous mutations.

2.2 Development and Modularity in Artificial Neural Networks

The brain is modular at several levels. The cerebral cortex, for example, contains macro-modules of hundreds of minicolumns, each module containing approximately one hundred neurons. There are many known examples of neural structures that serve a purpose different from their original use, for example [22]. When one considers animals as a whole, it is clear that most properties evolved from ancestral properties which had slightly different functions [16]. For example, if we could trace far enough back up our evolutionary tree, we would not expect ears to have suddenly appeared on an individual in a situation where ears would be useful. Similarly, we can expect most neural structures to be descended from neural structures which once had a different function.

Evidence from gene theory tells us that genes are used like a recipe, not a blueprint. In any one cell, at any one stage of development, only a tiny proportion of the genes will be in use. Further, the effect that a gene has depends upon the cell's local environment – its neighbours.

The above two paragraphs are related: For a type of module to be used for a novel function (and then to continue to evolve from there), without loss of current function, either an extra module must be created or there must be one 'spare' (to alter). Either way, a duplication system is required. This could be either by gene duplication or as part of a developmental process. Gene

duplication can be rejected as a sole source of neural module duplication, because our genes do not have the capacity to store all specific connections without a modular coding [1]. Therefore we arrive at the conclusion that a developmental process is required for the effective evolution of complex neural structures.

There are currently three main approaches to the modular development of artificial neural networks (ANNs): cellular encoding, cellular automata and Lindenmayer systems.

Before moving on to consider these three approaches, the type of neural network required should be considered. Most artificial neural networks that have been manually designed are layered feed-forward networks. However, recurrent networks can have internal state sustained over time and demonstrate rich intrinsic dynamics. This makes them attractive for use in adaptive behaviour work. Evidence from neuroscience provides further support, as biological neural networks are recurrent. Whilst recurrent ANNs can be very hard to study and construct manually, artificial evolution should have no problem using them. Indeed, there seems to be little reason to constrain the evolution to feed-forward networks, especially when aiming for autonomous agents that are to act as complex dynamical systems working within a time frame.

2.2.1 Gruau's Cellular Encoding

Gruau [8] uses genetic programming (GP) [14] to evolve his 'cellular programming language' code to develop modular artificial neural networks. The programs used are trees of graph-rewrite rules whose main points are cell division and iteration. Like many practitioners of GP, he considers evolutionary algorithms as a *tool* in the design process.

The crucial feature required for *this* work, which is missing from Gruau's approach, is exactly what is missing from genetic programming. Modularity can only come from either gene duplication (see objections above) or iteration. But iteration is not a powerful enough modular developmental backbone. Consider, for example, the cerebral cortex's macro-modules of hundreds of mini-columns. These are complicated structures that cannot be generated with a 'repeat one hundred times: mini-column' rule. There is variation between macro-modules.

So in genetic programming, we are reduced to gene duplication for all but simple iterative structures. What is required is a rule of the sort 'follow (rules X)' where (rules X) is a marker for (pointer to) rules encoded elsewhere on the genotype. But this would be difficult to incorporate into GP. A better route is to use a system capable of such rules. Both cellular automata and Lindenmayer systems are.

2.2.2 Cellular Automata

The use of conventional cellular automata (CA) for the construction of artificial neural networks has been recorded in many CA books. However, such work is more at the level of neuron growth than the development of whole networks. Although CA *rules* are suited to the evolution of neural network development in principle, the amount of work still to be done makes this a major research hurdle. Also, CA appear to offer no advantage for this work over the system chosen: Lindenmayer systems (next section), which are related to cellular automata.

CAM-Brain [7] is a project to "implement a cellular automata based artificial brain with a billion neurons by 2001, which grows/ evolves at (nano-) electronic speeds" (from the abstract). Whilst resulting technology may be of use in this field, there are two main problems with the CAM-Brain project. The first is that insufficient consideration has been given to the requirements of evolving intelligent behaviour. A single brain with no suitable means of interacting with other intelligences will be of little use for the evolution of intelligence. This could be overcome by using hundreds of these devices (possibly smaller and inside one machine). The second, more serious problem is that the CA *rules* are fixed. The method involves feeding a sequence of 'signal cells' along 'trails' (contained by 'sheath cells'). The sequence of signal cells is determined by the genotype. This amounts to a program and, without serious thought concerning how to incorporate a 'follow (rules X)' rule, warrants the same criticism as Gruau's cellular encoding.

2.2.3 Lindenmayer Systems

As discussed above, gene-theory has shown us that in nature genes are used as a recipe for each cell to follow. The development of each cell is determined by the relevant genes, which are determined by the cell's immediate environment. All cells use the same set of rules, derived from the genes.

Lindenmayer Systems [15] (L-systems) were developed to model the biological growth of plants. They are a class of fractals which apply rules ('production rules') in parallel to the cells of their subject. Thus a specified 'axiom' subject (typically one or two cells) develops by repeated re-application of these rules. Each step in a cell's development can be determined by the cell's immediate environment, including itself. In general, the most specific production rule that matches a cell's situation is applied.

For the purpose of this work, L-systems offer all the advantages of CA (and more) with none of the disadvantages. Also, work on the evolution of L-systems as modular development strategies for ANNs has already been undertaken with some success.

Kitano [13] used a context-free L-system to evolve connectivity matrices. The number of rules in the genotype was variable. After each developmental step, the ma-

trix would have doubled in both width and height. Kitano demonstrated better results than direct encoding when evolving simple ANNs (such as XOR and simple encoders) using training by backpropagation (gradient descent on the output error-vector's length). He also showed that the number of rules could be small. One criticism of Kitano's work must be that the resulting network architectures are fully-connected clusters of unconnected nodes; backpropagation is still doing most of the work.

Boers and Kuiper [1, 2] used a context-sensitive L-system to evolve modular feed-forward network architectures that were evaluated after training by backpropagation. A fixed-length alphabet was used for the rules, restricting the possible network architectures but still producing some good results.

The genetic decoding involved starting from each of the first five bits of the genotype and reading the string six bits (one character) at a time. Therefore each minimal substring that could be decoded into a production rule was read. Invalid production rules were thrown out. Then the decoding is repeated, using the genotype in reverse.

The evolution of production rules used a conventional genetic algorithm, with fixed-length genomes initially randomised. A limit of 6 rewrite passes over the network string was imposed (and reached by most genomes).

One criticism of Boers' and Kuiper's work must be the same as for Kitano's: that the resulting network architectures are fully-connected clusters of unconnected nodes; backpropagation is still doing most of the work. It is possible, from their results, that the genetic algorithm is doing little more than constructing layers of nodes with the provision for links that skip intermediate layers. However, good results were achieved on problems such as the TC-problem [18], and their work was the source of the main ideas behind the developmental system outlined in this paper.

3 Underlying Theory

3.1 Intelligent Artificial Life calls for Evolutionary Emergence

The (long-term) aim of creating systems with intelligence comparable to that in nature presents us with the problem that we do not understand such intelligence well enough to program it into a machine. Therefore we must either increase our understanding of it until we can, or create a system which outperforms the specifications we can give it. The first possibility includes traditional AI's top-down methodology, which has so far made little progress towards such intelligence. Ryle [19] long ago pointed out the futility of attempts to define intelligence, since they must invoke the 'ghost in the machine' fallacy: we can never observe intelligence directly; we can only define that some actions are more intelligent than others.

The first option also includes manual incremental (bottom-up) construction of autonomous systems with the aim of increasing our understanding and ability to model intelligence. The aim here is to build increasingly impressive systems, retaining functional validity by testing them within their destination environment. However, bearing in mind the fundamentally distributed nature of intelligence, it is unlikely that human designers will be capable of manually producing intelligence beyond a rudimentary level.

The second option is to create systems which outperform the specifications given them and which are open to producing behaviours comparable with those of (albeit simple) natural life. Specifically, the objectives are behaviours comparable with those of the only known intelligences: organisms with nervous systems as produced by evolution in nature. Thus the creation of evolutionary systems open to the emergence of intelligent behaviours surfaces as a promising route. It is my opinion that such emergent behaviours (however basic) could warrant the label 'real intelligence'.

When discussing this issue, the general-purpose nature of intelligence must not be forgotten. To argue that creating general-purpose intelligence is too vague or difficult a problem and that research should deal first with specific, static behaviours is analogous to advising Charles Babbage to think first about designing a machine to sort coins (say), before tackling the more demanding issues involved for a general-purpose computing machine. Such arguments completely miss the point. The way forward is to tackle the issue in its entirety.

3.2 Evolutionary Emergence calls for Natural Selection

Evolution in nature has no (explicit) evaluation function. Through organism-environment interactions, including interactions between (similarly-capable) organisms, certain behaviours fare better than others. This is why the non-random cumulative selection works without any long-term goal. It is why new abilities emerge. In order to achieve high levels of emergence, we must stop treating evolution as a search for the optimum organism as evaluated by some guess at what that involves. The use of evaluation functions results (at best) in solutions which score well but provides little freedom for emergent properties to progress.

As artificial evolution is applied to increasingly difficult problems, the difficulty in specifying satisfactory evaluation functions is becoming apparent (for example [24]). At the same time, the power of natural selection is being demonstrated in prototypal systems such as Tierra [17] and PolyWorld [23]. However, the total freedom of natural selection is still to be realised in an artificial life system. For example, Tierra's 'Darwinian operating sys-

tem' is used to kill old and defective programs, and PolyWorld uses a concept of energy which results in a fairly clear artificial evaluation function, albeit implicit.

3.3 Evolutionary Emergence of Intelligence calls for Developmental Modularity

The evolutionary emergence of intelligence requires new neural structures. We can expect most neural structures to be descended from neural structures which once had a different function [16]. For a neural structure to be used in a novel way (possibly after further evolution) and yet still perform its current roles, a duplication system is required. This could be either gene duplication or part of a modular developmental process.

Gene duplication can be rejected as a sole source of neural structure duplication, because the capacity required to store all specific connections in a large network without a modular coding is infeasible. Therefore, we conclude that for the effective evolutionary emergence of intelligence, a modular developmental process is called for. For the sake of research validity (regarding long-term goals), this should be included from the outset.

A modular development strategy should be able to use a module in a large number of specific situations (rather than just iteratively). It makes intrinsic sense that the development of a module should depend partly on its local environment, including itself, and not on remote parts of the environment. Lindenmayer systems [15] can possess both of these fundamental properties. Indeed, using the most general definition of an L-system, all systems with these properties are examples of L-systems.

4 The Artificial World

In order to develop and validate the underlying theory, a simple Artificial Life system has been created. 'Geb' (Egyptian god of the earth and father of Isis; also stands for Genetics and Emergent Behaviours) is a two-dimensional toroidal world containing artificial organisms each controlled by a neural network. The networks are produced from bit-string genotypes by a developmental process. Evolution within Geb is strictly by *Natural Selection*.

Geb's world is divided into a grid of squares; 20x20 of them in most runs. No two individuals may occupy the same square at any one time. This effectively gives the organisms a size within the world and puts a limit on their number (400 in most runs). Individuals are otherwise free to move around the world, restricted only by this simple rule. As well as a position within the world, each organism has a forward (facing) direction, set randomly at birth. Organisms are displayed as filled arcs, the sharp points of which indicate their forward direction.

This is Geb's main algorithm:

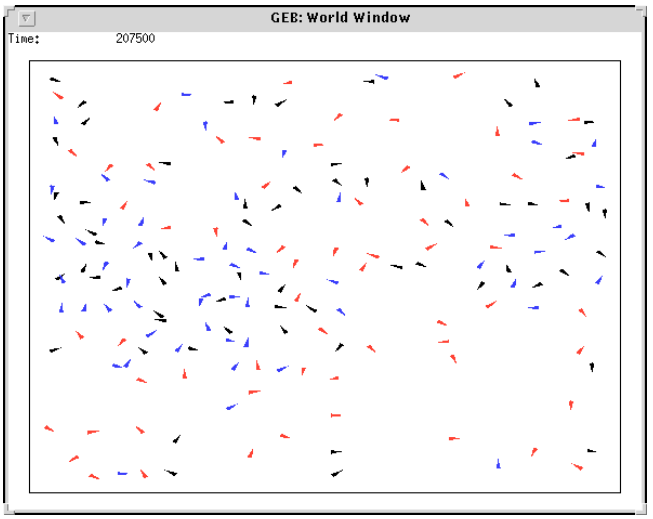

Figure 1: Geb World

Initialisation

Every square in the world has an individual with a single-bit genotype '0' born into it.

Main Loop

In each time step (loop), every individual alive at the start of the cycle is processed once. The order in which the individuals are processed is otherwise random.

These are the steps involved for each individual:

1. Network inputs are updated. See section 4.2.
2. Development – an iteration of the ontogenesis mechanism. See section 4.3.
3. All neural activations, including network outputs, are updated. See section 4.1.
4. Actions associated with certain network outputs are carried out according to those outputs. These actions are reproduce, fight, turn anti-clockwise, turn clockwise, and move forward. See section 4.2.

4.1 The Neurons

The artificial neural networks used in Geb are recurrent networks of nodes as used successfully by Cliff, Harvey and Husbands in their evolutionary robotics work [6, 11, 12]. Cliff et al. evolved recurrent networks of these nodes for visual navigation tasks in simple environments.

The level of noise used here (0.6) is significantly higher than that used by Cliff et al. (0.1). This is because noise is the only source of activation in Geb and, with the developmental method outlined below, it is easy for evolution to produce 'generator units' [12] (sources of high output). A corresponding (high) decision threshold for organisms' binary (yes/no) actions, such as reproduction, is used. Thus full control is available (via inhibition and generator units), early random binary actions are at

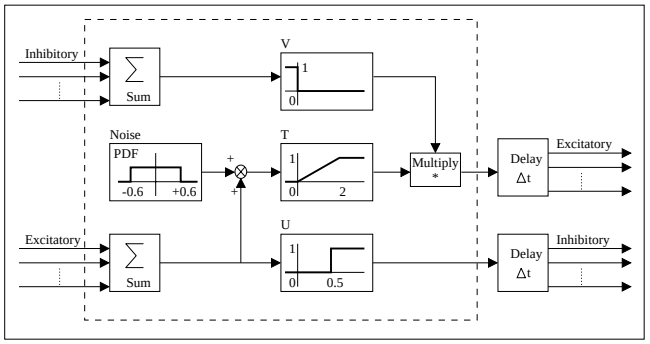

Figure 2: Schematic block diagram of a neuron, from [6]

a sensible level and early random multi-valued actions (such as moving forwards by a distance) can be at a reasonably high level without having to be scaled such that the maximum possible is ridiculously high. The neurons' veto-threshold is set to the decision threshold for organisms' binary actions. These details will become clearer as the rest of the system is described.

All links have unit weight; no lifetime learning is used. This is to avoid the criticism that lifetime learning is the main factor.

Each node has a bit-string 'character' (label) attached to it. This is used to match organisms' network inputs, outputs and actions, and to determine the node's development during the individual's lifetime. These characters may be of any non-zero length. A node may be a network input, a network output, or neither. This is determined by the developmental process.

4.2 Organism ⟷ Environment Interactions

There are five built-in actions available to each organism. Each is associated with network output nodes whose characters start with a particular bit-string:

1. `01*` Try to *reproduce* with organism in front
2. `100*` *Fight*: Kill organism in front
3. `101*` *Turn anti-clockwise*
4. `110*` *Turn clockwise*
5. `111*` *Move forward* (if nothing in the way)

For example, if a network output node has the character 1101001, then the organism will turn clockwise by an angle proportional to the excitatory output of that node. If an action has more than one matching network output node then the relevant network output is the sum of these nodes' excitatory outputs. If an action has no network output node with a matching character, then the relevant network output is noise, at the same level as in the (other) nodes.

Both 'reproduce' and 'fight' are binary actions. They are applied if the relevant network output exceeds a certain threshold and have no effect if the square in front is empty. Turning and moving forward are done in proportion to excitatory output.

When an organism reproduces with another in front of it, the child is placed in the square beyond the other individual if that square is empty. If the square is not, the child replaces the individual being mated with. An organism can not reproduce with an individual that is fighting if this would involve replacing the fighting individual.

Reproduction involves crossover and mutation. Geb's crossover always offsets the cut point in the second individual by one gene (bit position), with equal probability either way. This is why the genotype lengths vary. Also, crossover is strict. That is it always uses genes from both parents; the cut point cannot be at the very end of either genotype. This provides significant initial pressure for length increase until genotypes are long enough to produce developmental rules. Mutation at reproduction is a single gene- (bit-) flip on the child genotype.

An organism's network input nodes have their excitatory inputs set to the weighted sum of 'matching' network output nodes' excitatory outputs from other individuals in the neighbourhood. If the first bit of a network input node's character is 1 then the node takes its input from individuals to the right hand side (including forward- and back-right), otherwise from individuals to the left. A network input node 'matches' a network output node if the rest of the input node's character is the same as the start of the character of the output node. For example, a network input node with character 10011 matches (only) network output nodes with character's starting with 0011 in the networks of individuals to the right. The weights are inversely proportional to the Euclidean distances between individuals. Currently the input neighbourhood is a 5x5 square area centred on the relevant organism.

Notice that the network output nodes with characters $0, 1, 10, 11$ and all those starting with 00 do not produce any action. However, their excitatory values can still be input by other individuals. Thus there is the potential for data exchange not directly related to the actions.

4.3 The Lindenmayer Systems Used

A context-free L-system was designed for the evolution of networks of the neurons outlined above. Specific attention was paid to producing a system in which children's networks resemble aspects of their parents'.

Every node is processed once during each developmental step. The production rule that best matches the node's character is applied (if there is one). A rule matches a node if its predecessor is the start of the node's character. The longer the matching predecessor, the bet-

ter the match. Thus ever more specific rules can evolve from those that have already been successful.

The production rules have the following form:

$$\mathcal{P} \rightarrow \mathcal{S}_r, \mathcal{S}_n \ ; \ b_1, b_2, b_3, b_4, b_5, b_6 \quad \text{,where:}$$

$\mathcal{P}$	Predecessor (initial bits of node's character)
$\mathcal{S}_r$	Successor 1: *replacement* node's character
$\mathcal{S}_n$	Successor 2: *new* node's character
bits:	link details [0=no,1=yes]:
(b_1, b_2)	reverse types [inhibitory/excitatory] of (input, output) links on $\mathcal{S}_n$
(b_3, b_4)	(inhibitory, excitatory) link from $\mathcal{S}_r$ to $\mathcal{S}_n$
(b_5, b_6)	(inhibitory, excitatory) link from $\mathcal{S}_n$ to $\mathcal{S}_r$

If a successor has no character (0 length) then that node is not created. Thus the predecessor node may be replaced by 0, 1 or 2 nodes.

The 'replacement' successor (if it has a character) is just the old (predecessor) node, with the same links but a different character. The 'new' successor (if it has a character) is a new node. It inherits a copy of the old node's output links unless it has a link to the old node (b_5 or b_6). It inherits a copy of the old node's input links unless it has a link from the old node (b_3 or b_4).

New network input nodes are (only) produced from network input nodes and new network output nodes are (only) produced from network output nodes. The character-based method of matching up network inputs and outputs ensures that the addition or removal of a network input/ output node at a later stage of development or evolution will not damage the relationships of previously adapted network inputs and outputs.

The axiom network consists of three nodes with two excitatory links:

$$\text{network input } 001 \longmapsto 000 \longmapsto 01 \text{ network output}$$

These node characters were not chosen randomly. First, the network output node's character (01) matches reproduction. Next, the network input node's character (left input 01) matches this, without matching any of the other action characters. And finally the hidden node's character neither matches nor is matched by the other nodes' or the action characters.

Development takes place throughout the individual's life, although necessary limits on the number of nodes and links are imposed.

4.3.1 Context-Sensitive Lindenmayer Systems

To fully realise a developmental system satisfying the underlying theory (section 3.3), a context-sensitive L-system is called for. At the time of (re-)submission, experiments with such systems are underway and have produced the same initial results as those detailed in this paper but have not yet run for long enough to produce the later or different results.

4.4 Genetic Decoding of the Production Rules

The genetic decoding is loosely similar to that used by Boers and Kuiper [1, 2]. For every bit of the genotype, an attempt is made to read a rule that starts on that bit. A valid rule is one that starts with 11 and has enough bits after it to complete a rule; the number of bits varies because the node-characters can be of any length.

To read a rule, the system uses the idea of 'segments'. A segment is a bit string with its odd-numbered bits (1st, 3rd, 5th, ...) all 0. Thus the reading of a segment is as follows: read the current bit; if it is a 1 then stop; else read the next bit – this is the next information bit of the segment; now start over, keeping track of the information bits of the segment. Note that a segment can be empty (have 0 information bits).

The full procedure to (try to) read a rule begins with reading a segment for each of the predecessor, the first successor (replacement node) and the second successor (new node). Then, if possible, the six link-details bits are read. Only if this is achieved before the end of the genotype is a rule created.

For example:

```
Genotype: 1 1 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 0
Decoding: +++ ->_1_ * _0_ * 0 1 1 1 0 0
            +++ _1_ ->_0_ * _1_ * 1 0 0 0 0 0
Rules: 1.  P  -> Sr ,  Sn , link bits
          any -> 1  ,  0  , 0 1 1 1 0 0
       2.        P  -> Sr ,  Sn , link bits
                 1  -> 0  ,  1  , 1 0 0 0 0 0
```

After reading all possible rules from a newly-born's genotype, Geb filters the rules: It starts with rules whose predecessors best match a node in the axiom network, and then repeatedly adds in the best matching new rules if possible and as required to match predecessors to the successors of rules already picked. Rules that have not been picked when this process stops (because no new rules can be added under the criteria) have predecessors that could never match a node during development, at least not as well as another rule. In this way the redundant rules, which constitute the vast majority of decoded rules from long genotypes, are filtered out, much reducing the level of machine memory required by Geb.

During this process, a second criterion must be met for a rule to be added: the gene-segment the rule was decoded from must not overlap with a gene-segment of any rule already picked. This prevents the otherwise common situation of a rule $P{\rightarrow}R,N;bits$ producing successors R and N which can then be subject to rules $R{\rightarrow}N,B;C$ and $N{\rightarrow}B,C;D$ (and so on) as would be the case whenever P ends in 1 or R ends in 1.

5 Results

5.1 Kin Similarity and Convergence

When two Geb organisms (with networks developed from more than just a couple of production rules each) reproduce, the child's network almost always resembles a combination of the two parents' networks. This has been seen many times; Figures 3 to 5 are a typical example. These figures are shown in the form that Geb displays them on request, with all nodes crowded into three rows (network inputs, hidden nodes and network outputs). The 'pulling apart' and detailed examination of networks is left to be carried out by the user through the dragging and dropping of nodes.

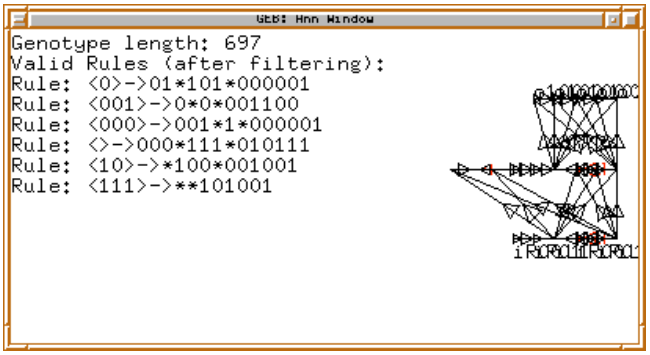

Figure 3: Parent 1

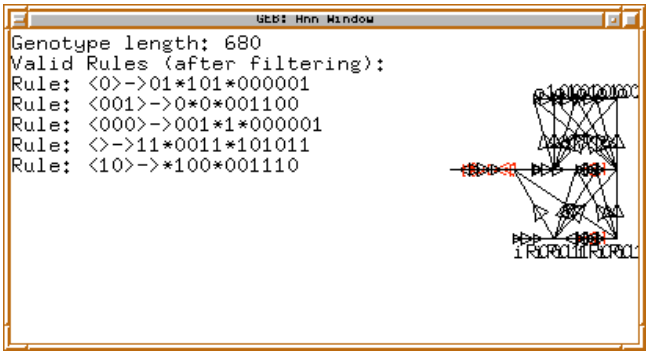

Figure 4: Parent 2

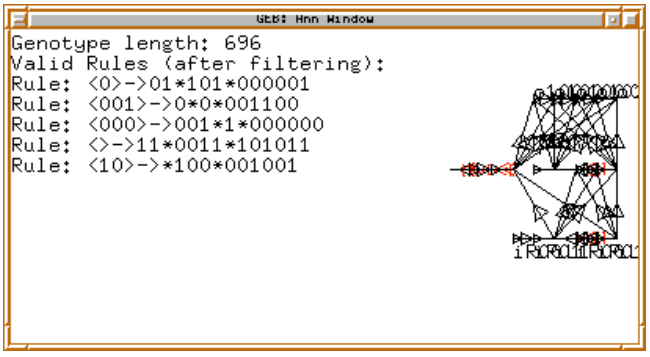

Figure 5: Child

Examination of a larger number of networks from Geb's population, at any time, shows similarities between many of the networks. The population remains nearly-converged, in small numbers of species, throughout the evolution.

5.2 Emergence of Increasing Complexity

Once Geb has started, there is a short period while genotype lengths increase until capable of containing a production rule. For the next ten to twenty thousand time steps (in typical runs), networks resulting in very simple strategies such as 'do everything' and 'always go forwards and kill' dominate the population. Some networks do better than others but not sufficiently well for them to display a dominating effect on the display of Geb's world window.

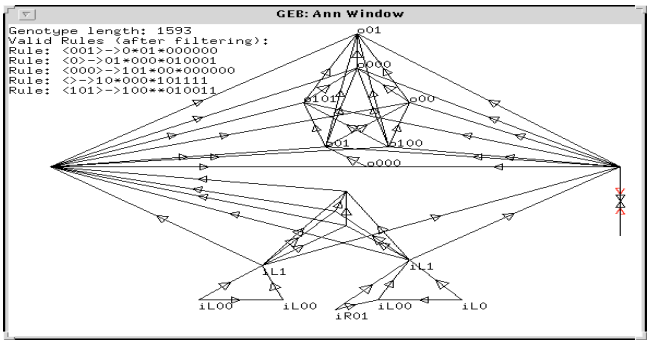

Figure 6: A Dominant Organism

In every run to date, the first dominant species that emerges has been one whose individuals turn in one direction while trying to fight and reproduce at the same time. Figure 6 shows an example of such an individual. Note the network outputs o101, o01 [x2] and o100 (turn anti-clockwise, reproduce and fight). Note also the large number of links necessary to pass from network inputs to outputs, and the network input characters which match non-action output characters of the same network (o000 [x2], o00). Individuals of this species use nearby members of the same species, who are also turning in circles, as sources of activation (so keeping each other going).

Although a very simple strategy, watching it in action makes its success understandable. Imagine running around in a small circle stabbing the air in front of you. Anyone trying to attack would either have to get their timing exactly right or approach in a faster spiral – both relatively advanced strategies. These individuals also mate just before killing. The offspring (normally) appear beyond the individual being killed, away from the killer's path.

Because of the success of this first dominant species, the world always has enough space for other organisms to exist. Such organisms tend not to last long; just about any movement will bring them into contact with one of

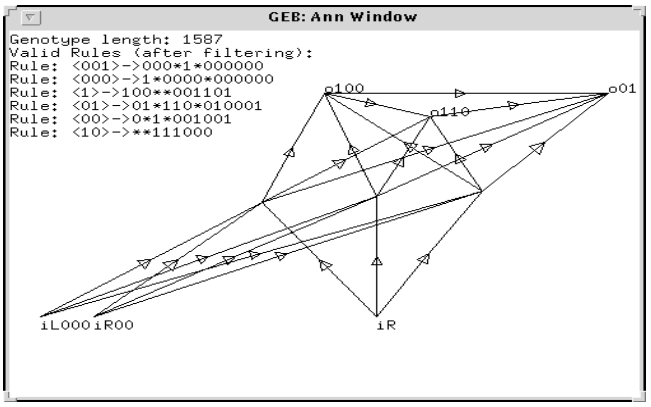

Figure 7: A rebel

the dominant organisms, helping that species in its reproduction as much as themselves. Hence these organisms share some of the network morphology of the dominant species. However, they can make some progress: Individuals have emerged that are successful at turning to face the dominant species and holding their direction while trying to kill and reproduce. An example of such a rebel (from the same run as figure 6) is shown in figure 7. Note that most rebels have many more nodes and links. This one was picked for its clarity.

Further, running averages of the number of organisms reproducing and killing (figures 8&9) show that a second dominant species tends to emerge, possibly evolved from the rebels. Such organisms have not yet been analysed.

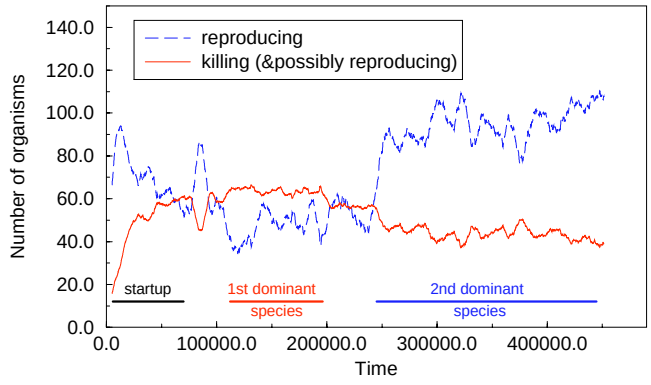

Figure 8: Typical run 1 (running averages)

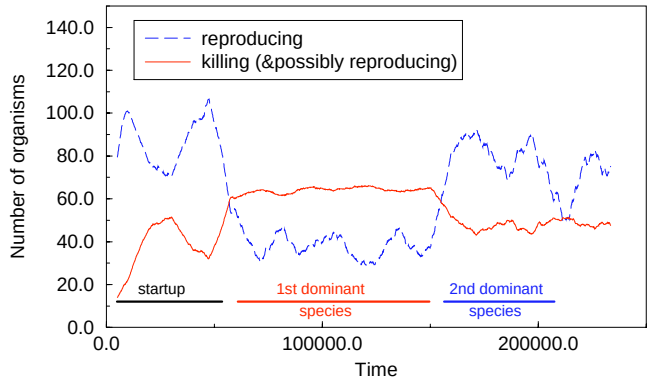

Figure 9: Typical run 2 (running averages)

5.3 An Unexpected Result

Whilst it was expected that inhibitory links would play an important role, the evolution in Geb has so far resulted in individuals with very few. This is despite the apparently high potential for inhibitory links in the developmental system.

6 Conclusions

The main conclusion is that the proposed approach is viable. Although the behaviours that emerged are very low-level, they are encouraging nonetheless, for they are similar in type to those found in nature and the increases in complexity were in ways not specified by the design.

In work involving pure natural selection, the organisms' developmental and interaction systems are analogous to the fitness functions of conventional genetic algorithms. Whilst the general aim involves moving away from such comparisons, the analogy is useful for recognising how the epistasis of fitness-landscape issue transfers across: Certain ontogenetic and interaction systems can result in individuals with similar genotypes but very different phenotypes. The results show that Geb does not suffer from this problem, probably because of the care taken in its design.

This work has made it clear that the *specification* of 'actions', even at a low-level, results in the organisms being constrained around these actions and so limits the evolution; natural selection alone does not guarantee an open-ended system. Viable alternatives in which the embodiment of organisms is connected to their actions need to be investigated.

7 Acknowledgements

This work is supported by an award from the EPSRC of Great Britain (supervisor Bob Damper, Southampton University). It is a continuation of previous work [5] also supported by an award from the EPSRC (supervisor Inman Harvey, Sussex University). My thanks to both supervisors for their encouragement in the area and comments on the work.

References

[1] Egbert J.W. Boers and Herman Kuiper. Biological metaphors and the design of modular artificial neural networks. Master's thesis, departments of Computer Science and Experimental Psychology, Leiden University, The Netherlands, 1992.

[2] Egbert J.W. Boers, Herman Kuiper, Bart L.M. Happel, and Ida G. Sprinkhuizen-Kuyper. Designing modular artificial neural networks. Technical report, Leiden University, 1993.

[3] Rodney A. Brooks. Intelligence without reason. In

Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), 1991.

[4] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[5] Alastair Channon. The evolutionary emergence route to artificial intelligence. Master's thesis, Cognitive and Computing Sciences, University of Sussex, 1996.

[6] Dave Cliff, Inman Harvey, and Phil Husbands. Incremental evolution of neural network architectures for adaptive behaviour. Technical Report CSRP256, University of Sussex School of Cognitive and Computing Sciences, 1992.

[7] Hugo de Garis. CAM-Brain. Technical report, Brain Builder Group, Evolutionary Systems Department, ATR Human Information Processing Research Laboratories, Kansai Science City, Kyoto, Japan, 1993.

[8] Frédéric Gruau. Artificial cellular development in optimization and compilation. Technical report, Psychology department, Stanford University, Palo Alto, CA, 1996.

[9] Inman Harvey. Species adaptation genetic algorithms: A basis for a continuing SAGA. In F.J. Varela and P. Bourgine, editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354, Cambridge, MA, 1992. MIT Press/Bradford Books. Also available as CSRP221, School of Cognitive and Computing Sciences, University of Sussex.

[10] Inman Harvey. Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In Christopher G. Langton, editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Vol. XVII*. Addison-Wesley, 1994. Also available as CSRP222, 1992, School of Cognitive and Computing Sciences, University of Sussex.

[11] Inman Harvey, Phil Husbands, and Dave Cliff. Issues in evolutionary robotics. In J.A. Meyer, H. Roitblat, and S. Wilson, editors, *Proceedings of SAB92, the Second International Conference on Simulation of Adaptive Behaviour*, Cambridge, MA, 1993. MIT Press/Bradford Books. Also available as CSRP219, School of Cognitive and Computing Sciences, University of Sussex.

[12] Phil Husbands, Inman Harvey, and Dave Cliff. Analysing recurrent dynamical networks evolved for robot control. In *Third IEE International Conference on Artificial Neural Networks (IEE-ANN93)*, pages 158–162, London, 1993. IEE Press.

[13] Hiroaki Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[14] John R. Koza. *Genetic Programming*. MIT Press/ Bradford Books, Cambridge, MA, 1992.

[15] Aristid Lindenmayer. Mathematical models for cellular interaction in development. *Journal of Theoretical Biology*, 18:280–315, 1968. Parts I and II.

[16] Ernst Mayr. The emergence of evolutionary novelties. *Journal unknown*, 1960s? In Southampton University Library (UK) ref.: WML:QH366MAY.

[17] Thomas S. Ray. Artificial life. In W. Gilbert and G.T. Valentini, editors, *From Atoms to Minds*. Istituto della Enciclopedia Italiana Treccani, Rome, October 1996. http://www.hip.atr.co.jp/~ray/.

[18] David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors. *Parallel Distributed Processing: explorations in the microstructure of cognition*, volume 1: Foundations. MIT Press, 1992.

[19] G. Ryle. *The concept of mind*. Hutchinson, 1949.

[20] A.V. Sannier II and E.D. Goodman. Genetic learning procedures in distributed environments. Technical report, A.H. Case Center for Computer-Aided Engineering and Manufacturing, Michigan State University, 1987.

[21] Luc Steels. The artificial life roots of artificial intelligence. *Artificial Life Journal*, 1(1):89–125, 1994. MIT Press.

[22] D.G. Stork, B. Jackson, and S. Walker. 'Non-optimality' via pre-adaptation in simple neural systems. In C.G. Langton, J.D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity, Vol. X*, pages 409–429. Addison-Wesley, 1991.

[23] Larry Yaeger. Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or polyworld: Life in a new context. In Christopher G. Langton, editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Vol. XVII*, pages 263–298. Addison-Wesley, 1994.

[24] Nahum Zaera, Dave Cliff, and Janet Bruten. (Not) evolving collective behaviours in synthetic fish. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96)*, pages 635–644. MIT Press Bradford Books, 1996.