Towards the evolutionary emergence of increasingly complex advantageous behaviours[1]

A.D. Channon and R.I. Damper

Abstract

The generation of complex entities with advantageous behaviours beyond our manual design capability requires long-term incremental evolution with continuing emergence. In this paper, we argue that artificial selection models, such as traditional genetic algorithms, are fundamentally inadequate for this goal. Existing natural selection systems are evaluated, revealing both significant achievements and pitfalls. Thus, some requirements for the perpetuation of evolutionary emergence are established. An (artificial) environment containing simple virtual autonomous organisms with neural controllers has been created to satisfy these requirements and to aid in the development of an accompanying theory of evolutionary emergence. Resulting behaviours are reported alongside their neural correlates. In a particular example, the collective behaviour of one species provides a selective force which is overcome by another species, demonstrating the incremental evolutionary emergence of advantageous behaviours via naturally-arising coevolution. While the results fall short of the ultimate goal, experience with the system has provided some useful lessons for the perpetuation of emergence towards increasingly complex advantageous behaviours.

Keywords: advantageous behaviour, complexity, emergence, evolution, natural selection

[1] This article appears in the Special Issue ‘Emergent Properties of Complex Systems’, 2000, *International Journal of Systems Science 31*(7), 843–860.

1 Introduction

This work aims towards the generation of systems within which increasingly complex advantageous behaviours can emerge. The adjective *advantageous* is used rather than *adapted* because the concern is the emergence of increasingly complex behaviours which mould a dynamical system of artificial entities, rather than just fit into an environment. This presents a dilemma: we do not understand such behaviours well enough to program them into a machine. So we must either increase our understanding until we can, or create a system which outperforms the specifications we give it.

The first possibility includes the traditional top-down methodology, which appears as inappropriate here as it has so far proved to be for (symbolic) artificial intelligence. It also includes most manual incremental (bottom-up) construction of autonomous systems with the aim of increasing our understanding and ability to model complex behaviours. The goal here is to build increasingly impressive systems, retaining functional validity by testing them within their destination environments (e.g. Wilson 1991). However, by the very nature of complexity, it is unlikely that human designers will be capable of manually producing complex *advantageous* behaviours beyond a rudimentary level.

The second option is to create systems which somehow outperform the specifications given them and which are open to producing increasingly complex advantageous behaviours. Evolution in nature has no (explicit) evaluation function. Through organism-environment interactions, including interactions between similarly-capable organisms, certain behaviours persist while others die off. This is how the non-random cumulative selection operates without any long-term goal; it is why novel structures and behaviours emerge. Whereas work on adapted behaviour focuses on fitness in the present, research on advantageous behaviour shifts the focus to the future, where what it is to be fit may have changed because of the moulding of the environment. However, this does not prevent us from evolving advantageous behaviours and discovering which ones are fit with respect to the resulting evolving environment – that is, which ones persist. Further, one can expect many behaviours that evolve and persist to have done so because they are (or at least have been) advantageous, in that the behaviours contribute to the persistence of their host system (organism, species or such).

This paper presents a detailed argument for the use of natural selection systems as a means of generating evolutionary emergence, before describing experimental results which further develop a methodology for constructing such systems. Accordingly, the remainder of the paper is structured as follows. Section 2 specifies what evolutionary emergence is, and how it differs from other types of emergence. Section 3 summarises incremental artificial evolution theory, necessary for long-term evolutionary emergence. Section 4 makes the case that artificial selection cannot generate evolutionary emergence (by our definitions), which must therefore be the product of natural selection. Section 5 evaluates existing natural selection systems. This section is more than a tutorial: it offers important and novel insights into these systems, revealing not only their significant achievements but also crucial pitfalls. Thus, it provides the first major contribution of this paper. Section 6 addresses the issue of *what* to evolve, the focus being on principles of neutrality and neural developmental modularity that enable the incremental evolution of complex behaviours. The experimental system, which satisfies the requirements identified thus far, but which also reveals further crucial pitfalls, is described in section 7. The purposes of this system are to verify and extend the theory of evolutionary emergent system generation. Section 8 presents the results and section 9 concludes, summing up what has been learned from the system – the second major contribution of this paper.

2 Evolutionary emergence

According to Stephan (1998, p. 639): "In different disciplines such as philosophy of mind, dynamical systems theory, and connectionism the term 'emergence' has different jobs to perform." For the purposes of this paper, we take emergence to be related to qualitatively novel structures and behaviours which are not reducible to those hierarchically below them. Thus, it offers an attractive methodology for tackling Descartes' Dictum: "how can a designer build a device which outperforms the designer's specifications?" (Cariani 1991, p. 776). Most important, it is *necessary* for the generation of complex entities with behaviours beyond our manual design capability.

Cariani identified the three current tracts of thought on emergence, calling them "computational", "thermodynamic" and "relative to a model". Computational emergence is related to the manifestation of new global forms, such as flocking behaviour and chaos, from local interactions. Thermodynamic emergence is concerned with issues such as the origins of life, where order emerges from noise. The emergence relative-to-a-model concept deals with situations where observers need to change their model to keep up with a system's behaviour. This is close to Steels' (1994) concept of emergence, which refers to ongoing processes which produce results invoking vocabulary not previously involved in the description of the system's inner components – "new descriptive categories" (section 4.1).

Evolutionary emergence falls into the emergence relative-to-a-model category. An example will clarify the divisions. Consider a virtual world containing organisms that can move and try to reproduce or kill according to rules which are sensitive to the presence of other organisms and which evolve under natural selection. Should flocking manifest itself in this system, we could classify it as emergent in two senses: first in the computational sense from the interaction of local rules, flocking being a collective behaviour, and second in the relative-to-a-model sense through the evolution, the behaviour being novel to the system. Although the first sense is also relevant to our goal, in that complex advantageous systems will involve such emergence, the second is the key to understanding *evolutionary* emergence.

Langton (1989) gave a simple, compatible method of ascribing emergence: "The essential features of computer-based Artificial Life models are: ... There are no rules in the system that dictate global behavior. Any behavior at levels higher than the individual programs is therefore emergent" (pp. 3–4). Note that this can be used for both the computational and relative-to-a-model senses of emergence. He also stressed (p. 41) the importance of nonlinear systems – those which do not obey the superposition principle (i.e. which cannot be understood in terms of independent constituent parts) where it is necessary to understand the interactions between the parts. Thus, new descriptive categories cannot be invoked from a system which obeys the superposition principle.

Having specified what is meant by evolutionary emergence, we will now summarise *incremental* artificial evolution theory and then explore the two types of selection which might be used to bring about such emergence. Packard (1989) referred to these as "*extrinsic* adaptation, where evolution is governed by a specified fitness function, and *intrinsic* adaptation, where evolution occurs 'automatically' as a result of the dynamics of a system caused by the evolution of many interacting subsystems" (p. 141). We will use the terms artificial and natural selection respectively, because the first involves the imposition of an artifice crafted for some cause external to a system beneath it, while the second relies solely on the innate dynamics of a system. Ray (1996, section 2.1) is one of the better known personalities trying to bring an awareness of the difference between artificial and natural selection to the many practitioners in the artificial evolution field who claim to be using natural selection when they are in fact using artificial selection.

3 Incremental artificial evolution

Genetic algorithms (GAs) are biologically inspired search procedures initially developed by Holland (1962, 1975, 1992) in the early 1960s – although see also Fraser (1957), Fogel (1962) and Fogel, Owens, and Walsh (1966) for other evolutionary algorithms' roots. GAs evolve an initial random population of genomes (codings for solutions to the problem in hand) by selecting which individuals are reproduced and which are replaced. This is done by evaluating each solution's fitness via some function relevant to the problem and favouring the fitter solutions. Reproduction typically involves both crossover, whereby parent genomes are split into sections at common (randomly chosen) cut points and the new genome inherits corresponding sections from one parent or the other, and mutation, which involves randomly altering a small proportion of the new genome. Mutation increases diversity and crossover combines beneficial discoveries. There are many variations on the typical GA, but most share this base description.

Although most GAs work on populations of solutions with a fixed size and structure, the evolution of increasingly complex entities requires us to evolve variable-sized genotypes over many generations. Harvey's (1993b) Species Adaptation Genetic Algorithm (SAGA) theory provides a framework for incremental artificial evolution. In this paradigm, a population (with possibly just a few tens of members) evolves as nearly-converged species, for thousands or millions of generations.

The increases in complexity must therefore result from the evolution itself. This is in contrast to the common use of the genetic programming (GP) paradigm (Koza 1990, 1992), for example, where a population of millions may be evolved for less than a hundred generations (Harvey 1997, section 5). In the GP case, recombination effectively mixes the random initial population, exhausting variation in few generations. Because genetic codings of computer program instructions result in rugged (uncorrelated) fitness landscapes (i.e. mutating a bit in the genotype of a fit program will almost certainly produce a very unfit program), there can be little further evolution of this converged population. Here we see one of the requirements of SAGA: a sufficiently correlated fitness landscape (actual or implicit). Mutation must be possible, at a low rate, without dispersing the species in genotype space or hindering the assimilation by crossover of beneficial mutations into the species.

The open-ended evolution of increasing complexity cannot, of course, be achieved with fixed-length genotypes. Harvey (1992) states the case for gradual changes in genotype length (sections 2–6). First, he reports some theory due to Kauffman and Levin (1987). In an adaptive walk on a completely uncorrelated landscape, with "fitness achieved" defined as the highest encountered, each step up the fitness rank will (be expected to) take twice as long as the previous step, there being only half the (expected) number of fitter neighbours. This result still holds if in each time step a large population (of fixed number) samples different mutants, with the population moving as a whole to the fittest. Harvey (1992) makes the point that the result holds on correlated landscapes for a "long jump ... defined to be the equivalent of several *simultaneous* mutations, long enough to jump beyond the correlation lengths in the landscape" (section 5). Thus, the argument proceeds, such long jumps will play less and less of a beneficial role as evolution progresses. After an initial period of fluctuation, only small jumps (such as individual mutations) will be beneficial. Harvey notes that this is not connected to the punctuated equilibria controversy; only a single large step is ruled out, not a cascade of small steps that could be rapid in geological time.

Harvey (1993a, section 6.7) provides an accurate tail of this argument. A change in genotype length which causes the information content of the genotype expressed in the phenotype (GIP) to change can also be considered as a mutation in the above argument. Therefore, such changes would only be beneficial if made in small jumps. Because detrimental mutation must not be so high as to disperse members away from existing fit genotypes, the argument is now complete for a low rate of change in the information content of the GIP, both by direct genetic mutation and indirectly through changes in genotype length.

If this rule – of a low rate of direct genetic mutation and a low rate of change in genotype length with respect to effect on information content of GIP – is followed, then the population will evolve as nearly-converged species, convergence being in terms of GIP information content. If, in addition, a direct genetic coding (i.e. a bijection between genotype and phenotype) is used, then the population's members will have an almost uniform genotype length that increases in small steps. Even if this is not the case, the information content of the GIP will increase (at most) gradually.

As made explicit by Harvey (1992, figure 3), the convergence of the population need not be around a single species. Variation in number of species is not engineered in, but rather is 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. Note also that the "problem of premature convergence" from traditional GA theory is now irrelevant.

One further issue worth clarifying is that of functional validity with respect to the destination environment, or "situation within a world". Brooks (1991a, 1991b) argues that incremental development (including evolution) must take place within the environment that the objects inhabit. This is to avoid problems (common in traditional artificial intelligence) caused by a divide between a system and the real world. So, for example, some researchers (e.g. Harnad 1993) argue that robots intended to inhabit the real world must be evolved (or at least frequently evaluated) in it. However, if organisms are only ever to inhabit an 'artificial' environment then there should be no concern about them being evolved in that environment. Their 'world' is not a simulation and so the approach suffers none of the problems that occur when trying to use a simulation to evolve robots for the real world. Where the artificial environment differs from our world (however greatly), there is no problematic error. There is simply a difference.

4 Artificial selection

Holland did not originally envisage GAs as functional optimisers, but rather as processes similar to natural adaptive systems. In the natural world, organisms interact in complex ways and so coevolve with their environment, which includes other organisms. However, GAs proved suitable for a range of optimisation tasks and this has grown to be their most widespread application (Goldberg 1989).

4.1 The state of artificial selection work

Within the artificial evolution field, variants of the optimisation paradigm have proven fruitful. Even where the concepts of SAGA theory (section 3) are dominant, practice still holds to the use of fitness functions. But as the complexity of behaviours under consideration increases, flaws in the artificial selection approach are appearing. Zaera, Cliff, and Bruten's (1996) failed attempts at evolving schooling behaviour in artificial fish provide an account of the difficulties faced:

> "The problem appears to be due to the difficulty of formulating an evaluation function which captures what schooling is. We argue that formulating an effective fitness evaluation function for use in evolving controllers can be at least as difficult as hand-crafting an effective controller design. Although our paper concentrates on schooling, we believe that this is likely to be a general issue, and is a serious problem which can be expected to be experienced over a variety of problem domains."

Zaera et al. considered possible reasons for their failure. The argument which most convinced them was that real schooling arises through complex interactions, and that their simulations lacked sufficient complexity (their section 5). They cited two promising works: Reynolds' (1992) evolution of coordinated group motion in prey animats pursued by a hard-wired predator, and Rucker's (1993) ecosystem model in which Boid-like animat controllers (or rather their parameters) were evolved. Both of these are moves towards more intrinsic, automatic evolution.

The use of coevolutionary models is fast becoming a popular approach in the adaptive behaviour field. This is essentially a response to the problems encountered when trying to use artificial selection to evolve complex behaviours. However, artificial selection has kept its hold so far – most systems still use fitness functions. Much of this work is based on the "Red Queen" or "Arms Race" phenomenon (see Cliff and Miller 1995 and Dawkins and Krebs 1979), an early example of which is Hillis' (1990) coevolution of sorting networks and their test cases. Hillis concluded his paper with the statement: "It is ironic, but perhaps not surprising, that our attempts to improve simulated evolution as an optimisation procedure continue to take us closer to real biological systems" (p. 233).

As with Hillis' paper, the reason given for imposing coevolution is often that it provides "a useful way of dealing with the problems associated with static fitness landscapes" (Bullock 1995, section 5). It appears that few of those working with artificial selection intentionally use coevolution as a step towards intrinsic evolution. Notably, Reynolds (1994) of Boids fame worked towards more automatic evolution by coevolving simulated mobile agent controllers which competed with each other in games of Tag. This eliminated the need to design a controller in order to evolve a controller, as in his previous work (Reynolds 1992) mentioned above.

4.2 (No) Emergence via artificial selection

From the above discussion, one might imagine our argument to be developing toward the extreme statement that evolutionary emergence is not possible in a system using artificial selection. This is not quite so, although we do argue that artificial selection is not sufficient. We now give an example of emergence from a genetic algorithm which Ray (1996) would classify as using "partial natural selection" (section 2.8), in that the interactions between artificial entities play a role commensurate with the artificial fitness function aspect of selection.

Sannier and Goodman (1987) used a distributed GA to evolve genomes within a two-dimensional toroidal grid containing "food" which is placed into the environment according to some pattern. An individual's "strength" (fitness), which is deducted from its parents' strengths at birth, increases on consumption of food and decreases in each time step (and upon reproduction). An individual is reproduced if its strength is above a threshold, and killed if its strength drops below a lower threshold. A genome encodes rules which allow it to move in eight directions (N, NE, E, SE, . . .) with program branching conditional on the presence of food in the eight neighbouring locations. Thus, the individuals can interact (only) by moving around and consuming food, so affecting each other's program branching.

In the experiment reported, food was restricted to two farm areas, spaced apart in the toroidal world. The level of food introduced into the farms was 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: *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 increased as it went through a farm and decreased 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. While farmers could survive without nomads, it was found that nomads needed farmers so that the farms would not be neglected between visits.

The important feature is the emergence of the two classes of individual. Never was it specified that they should arise. Evolution produced them because they perform better than other genomes within the environment. The previous paragraph demonstrates the need to update our model to include the new descriptive categories.

It would, of course, be fraudulent on our part to claim that this is an example of emergence via artificial selection. In this partial natural selection system, the artificial component of selection is incidental to the emergence, the source of which is the natural component of selection arising from interactions of the system's parts. The statement that "artificial selection is not sufficient [for evolutionary emergence]" (from the first paragraph of this subsection) does not imply a necessity for (or even benefit of) artificial selection. *In the context of evolutionary emergence, any artificial selection used constitutes just one of the parts of a system.*

In summary, artificial selection can only select for that which is specified. Therefore anything that emerges during evolution (in the evolutionary emergence sense) *must* result from another aspect of selection, which must in turn arise from the innate dynamics of the system – natural selection.

Artificial selection can result in evolved solutions that an experimenter had not anticipated. For example, a highly fit solution might only use a fraction of the genotype that had been made available and thought necessary, or a solution might exploit properties of the phenotype that had previously been unknown. However, examples such as the former do not require an extension to the observer's model, and examples such as the latter require an extension that is the product not of evolution, but of the observer's lack of knowledge. Neither qualifies as evolutionary emergence by our definition.

5 Natural selection

Natural selection retains the fitter individuals (those that persist) without any explicit specification of what is required to be fit, which changes as the system evolves. This feedback in the selection process is the vital factor missing from purely artificial selection systems (figure 1).

As noted in section 3, genetic codings of computer program instructions result in rugged fitness landscapes and this makes them unsuitable for incremental evolution by artificial selection. One would expect this argument to carry through to natural selection systems, where fitness is an abstract concept or external measure. However, the approach of most natural selection work to date has been to evolve program code, following the initial success of Tierra (Ray 1991) which demonstrated incremental evolution over millions of reproduction cycles. Despite a lack of continuing emergence, this early success needs to be explained against the argument that computer programs are not suitable for incremental evolution.

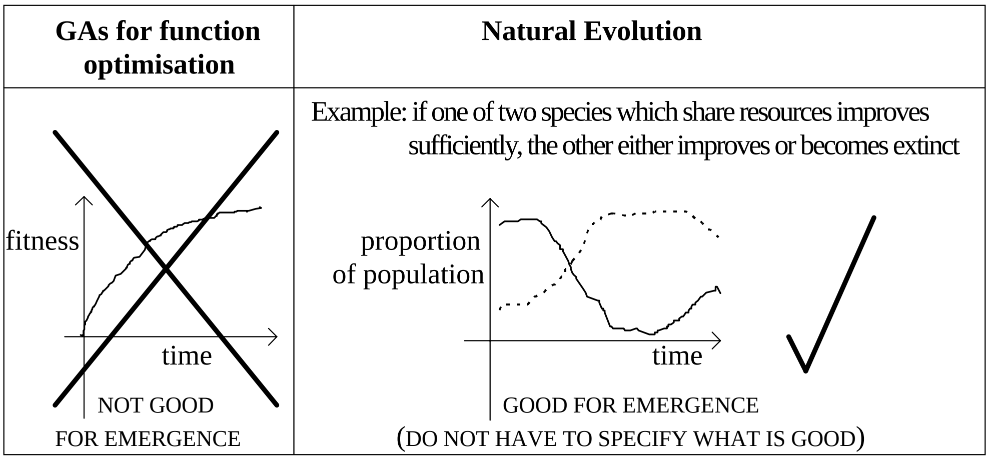

GAs for function optimisation
Natural Evolution
Example: if one of two species which share resources improves sufficiently, the other either improves or becomes extinct
fitness
time
NOT GOOD FOR EMERGENCE
proportion of population
time
✓
GOOD FOR EMERGENCE
(DO NOT HAVE TO SPECIFY WHAT IS GOOD)

Figure 1: The difference between artificial and natural selection.

5.1 Natural selection of program code

Tierra is a system of self-replicating machine code programs. As an evolutionary biologist, Ray was interested in comparing artificial evolution with that in the real world. To make evolution possible, a certain rate of random bit-flipping was imposed on the memory within which the population of running code resided. Each program was allowed to write to the block of memory it occupied but not outside that block. However, programs could read and execute instructions from any part of memory.

The population was initialised as a single manually-designed, self-replicating program. This program first examined itself to determine its length in memory, then issued an instruction to allocate free memory for a child, copied its code byte by byte to this free memory and finally issued an instruction to treat the child as an independent process.

A degree of artificial selection was imposed by the system itself deleting the oldest programs to free memory when it was full beyond a certain threshold, with an added bias against programs that generated error conditions. As emphasised at the end of section 4, this artificial selection constitutes just one of the parts of the system. It does not (necessarily) prevent natural selection.

Tierra was implemented as a virtual computer, allowing Ray to design a machine language with some properties suiting it to evolution. One aspect of this language was that it contained no numeric constants. This was to reduce the brittleness of the language by decreasing the size of the "real" instruction set, in which `add 1` and `add 2` are considered to be distinct instructions. Thus, direct memory addressing was not possible, in either a relative or absolute form. Instead, the manually-designed program began and ended with consecutive `NOP`s (No-OPerations) which acted as templates that could be found by certain machine code instructions which search memory backwards or forwards in a single step. This addressing by templates is how the program examined itself to determine the points at which to begin and end copying and so also its length.

Computational errors were introduced at random. For example, a left bit-shift instruction would sometimes shift a register's bits two positions, sometimes not at all. A copy instruction would occasionally copy to a neighbour of the correct location. Such errors could lead to genetic changes by affecting replication.

When Tierra was run, various classes of programs evolved. "Parasites" had shed almost half of their code, allowing them to replicate almost twice as fast; they replicated by executing the copy loop from neighbouring organisms, which could easily be found by template matching instructions as before. Because the parasites depended on their "hosts", they could not displace them and the host and parasite populations entered into Lotka-Volterra population cycles, characteristic of predator–prey and parasite–host systems (Lotka 1925; Volterra 1926). Ray reported that coevolution occurred as the hosts became immune to the parasites, which overcame these defences, and so on. "Hyper-parasite" hosts emerged containing instructions that caused a parasite to copy the host rather than the parasite; this could lead to the rapid elimination of the latter. Ray also reported cooperation (symbiosis) in replication followed by “cheaters” (social parasites) which took advantage of the cooperators.

The above are examples of ecological adaptations, which involved interactions between the programs. Another class of adaptations found was optimisations, where individual programs replicated faster than their ancestors. For example, non-parasitic replicators almost a quarter the length of the initial replicator were found, as were programs with unrolled copy loops which copied two or three bytes per loop, reducing the overhead of looping. By adding split and join instructions, which allowed a program to split into a multi-threaded process and join back into a single one, the evolution of efficient parallel-processing replicators was later achieved.

While the results of Tierra are impressive, the system is not truly open-ended. There have been no new reports of emergent phenomena during the last few years and it is generally accepted that not much more will occur unless further alterations are made to the system, as with the addition of the split and join instructions. Indeed, Ray (1996) is currently establishing a “biodiversity reserve for digital organisms” (section 5.2) based on a networked version of Tierra, in an attempt to generate more complex organisms. His hope is that the increased scale will hold an ecological community of many species, with the network model providing initial selective forces resulting from its temporal and spatial complexity. “Once a significant impulse in the direction of complexity has occurred, the hope is that selective forces arising from interactions among the digital organisms can lead to an auto-catalytic increase in complexity” (section 5.2.2). However, this system has not yet produced any encouraging results – see Ray (1997).

The evolvability of the code seems to stem largely from the template matching system. This could account for all of the ecological adaptations reported but would be of little use for much other than replication. To see how this could be, consider the pseudocode of the initial, manually-designed algorithm (1).

```
T1111 a=address(T1110)+1 ; b=address(T1111) ; c=a-b
T1101 allocate memory for child, length c =>a=start
      call T1100 ; divide from child ;jump to T1101
T1100 push a,b,c onto stack             ⎫
T1010 memory(a)=memory(b)               ⎪ copy
      c-- ; if c=0 then jump to T1011   ⎬ loop
      a++ ; b++ ; jump to T1010         ⎪
T1011 pop c,b,a off stack ; return      ⎭
T1110
```

Algorithm 1: Pseudocode for Tierra’s initial, manually designed program.

This pseudocode is based on the initial program as listed in Ray (1992, appendix C). The `T????s` on the left denote four-bit templates (with the same bits as the real templates), which can be thought of as labels; the difference is that a jump or call instruction will search outwards in memory to find the nearest matching template. Now, as reported by Ray (1992, section 3.1.1), a parasite can be obtained by simply mutating one bit of the `T1100` template to produce `T1110`; this would then be the same as the end template, reducing the length to be copied (`c`), and the `call T1100` statement would search outwards in memory until it found a host containing the copy loop. Further, a host that is immune to such a parasite can also be produced by a single-bit template mutation of the initial program. To be more exact, a single-bit mutation in the template-comparison `1011` in the `if c=0 then jump to T1011` statement of the copy loop achieves this; by mutating it to `1111`, the program will re-evaluate its address and length after every reproduction. Thus, should a parasite try to use this program’s copy routine, it will be copied just once but ever after that it will be copying the host; this host is also a hyper-parasite as defined above. In just two template bit-flips, we have reproduced the most impressive ecological results of Tierra! While the actual evolution might have taken a slightly different route, to slightly different programs, these phenomena clearly result from the flexibility of the template matching. It is also clear that the same argument applies to the other ecological adaptations reported.

Although this demonstrates the flexibility of template matching at the four-bit level, it also shows the ecological results to be somewhat less dramatic than first impressions suggest. Complex programs would necessarily contain many more templates (labels) and so these templates would have to be significantly longer. Thus, we would pass far beyond the simple four-bit templates that random search can operate on, to a stage where evolution is impractical. So, Tierra no longer constituting evidence to the contrary, we hold to our previous argument that programs are not suitable for (long-term) incremental evolution. Note that we are not dismissing Ray's work as trivial or unimportant; we consider Tierra to be a significant milestone in the artificial evolution field – the first intentional example of *naturally arising* coevolution in an artificial evolutionary system.

There is still the issue of the optimisation adaptations to address. If programs are too brittle for (long-term) incremental evolution, then how was it that these adaptations evolved? Two of the results, unrolled copy loops and efficient replication by parallel processing, can be easily explained. All evolution needed to do was insert (probably by the action of the random computational errors) repetitions of neighbouring code; the local functionality was not changed and the efficiencies are from more-of-the-same solutions. These examples should not alter our perception of the brittleness in mutating code or inserting *different* instructions. The final results to be explained are the non-parasitic replicators almost a quarter the length of the initial replicator. Comparing the shortest self-replicating program (Ray 1992, appendix D) with the initial program (appendix C) shows that the transition can be made by simply deleting instructions (mostly `NOP`s – cutting out the redundancy in the templates) and just six mutations, three of which are completely unnecessary. So apart from three mutations, this is a less-of-the-unnecessary solution, which is insufficient to challenge our argument.

Computer Zoo (Skipper 1992), inspired by Ray, shared most of the features of Tierra and demonstrated very similar results. Skipper noted that remote execution is essential to the evolution of parasites and hyper-parasites. There have been many other Tierra-inspired systems, such as Avida (Adami and Brown 1994) and Cosmos (Taylor and Hallam 1997), all of which failed to produce significantly more impressive results. There are other possible explanations for this, including an inability on our part to detect further evolutionary emergence, and/or a deficiency in the language that lies not in its brittleness but in its lack of expressive ability within the system. However, we believe that the discussion presented here eclipses such possibilities, in that it provides a sufficient explanation.

One piece of work often cited as an example of evolving programs by natural selection is that of Koza (1993). We omit a full discussion here because, although natural selection is possible within the presented framework, the improvements reported result from artificial selection involving evaluation at a task. The only example of emergence reported was the existence of 604 self-replicating programs in a set of 12,500,000 randomly generated programs. There was no emergence resulting from evolution.

5.2 The evolution of more suitable entities via natural selection

Although the approach of most natural selection work to date has been to evolve program code, there are two notable exceptions. Both involve the evolution of neural networks, which are well suited to incremental artificial evolution because of their graceful degradation (high degree of neutrality). The second (chronologically) is the work described in this paper, which was conceived (Channon 1996) independently of and initially created in ignorance of both Tierra (including its derivatives) and the first exception: PolyWorld (Yaeger 1993). Yaeger stated three motivations for his work, including the production of emergent complex behaviours and the exploration of artificial life as a route to artificial intelligence. These are (very) similar to the motivations behind the work presented here. However, his other motivation is what sets PolyWorld's design apart; it was "to create *artificial* life that is as close as possible to *real* life, by combining as many critical components of *real* life as possible in an *artificial* system" (p. 264). Hence PolyWorld simulates many aspects of the real world including energy conservation, food, movement (on a 2D plane), vision, neural networks and learning.

PolyWorld organisms have seven pre-programmed behaviours: eating, mating, fighting, moving, turning, focusing and lighting. These are expressed according to the activation levels of pre-specified neurons. An organism's chromosome determines its physiology (size, strength, maximum speed, green coloration, mutation rate, number of crossover points and life span) and some basic characteristics of its neural network. These characteristics include the number of neurons devoted to vision (in three groups – red, green and blue), the number of internal neuronal groups, a connectivity density (between groups) matrix and Hebbian learning rates. The neural networks are constructed stochastically.

PolyWorld is initially seeded with random genomes and run as a steady-state GA using an *ad hoc* fitness function until a population is achieved that maintains its number through mating. This occurred in the seed population of some runs, yet not at all in others. Yaeger reported a variety of emergent behaviours, including some trivial ones but also “fleeing”, “fighting back”, “grazing”, “foraging” and “following”. He also claimed that a drifting group of organisms and “one example of a few organisms [apparently] ‘chasing’ each other were even suggestive of simple ‘flocking’ behaviours” (p. 285), although this seems somewhat speculative.

One criticism of PolyWorld, in the context of perpetual evolutionary emergence, is that (Hebbian) learning appears to be overwhelmingly responsible for the results. There is little evidence of significant *genetic* evolution; the genomes had very limited control over the small number of neural groups. It is conceivable that if comparison PolyWorld experiments were run with birth networks specified by random parameters (or perhaps constant parameters which result in large enough neural networks), then the same results might emerge. New organisms would either learn such that they become adapted to the evolving environment or die, and so evolution would occur without genetic change. Although this is valid evolutionary emergence, it is not sufficient for perpetuating evolutionary emergence. Unless learning is made evolvable, or what is learned can be passed on, a maximal level will be reached at which organisms are not capable of learning more in their lifetimes.

Channon’s own experimental system – detailed here in full, but see also Channon (1996), Channon and Damper (1998a) and Channon and Damper (1998b) for the development of the ideas behind it – shares many features with PolyWorld, despite having been conceived without knowledge of Yaeger’s work. However, because it does not attempt to simulate aspects of the real world, it is considerably simpler. Most important, there is no learning in the neural networks, purposely to avoid the criticism above. Further, there is no obvious implicit fitness function, such as the “energy” in PolyWorld, that might dominate selection. The results reported here are comparable with (but do not exceed) those of PolyWorld, despite this removal of learning.

6 Developmental requirements

Natural selection is necessary for evolutionary emergence but does not guarantee the evolution of evermore novel emergent phenomena. The question “what class of objects can / should we evolve?” needs to be answered with that in mind, along with the central aim: increasingly complex advantageous behaviours. Neural networks are the clear choice because of their graceful degradation (as noted in section 5.2) and suitability for this aim. But how should the network structure be specified?

The evolutionary emergence of novel behaviours requires new neural structures, or “modules”. We can expect most to be descended from neural structures which once had different functions (Mayr 1960). There are many known examples of neural structures that serve a purpose different from a previous use, for example Stork, Jackson, and Walker (1991).

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 structure duplication, because the capacity required to store all connections in a large network without a modular coding is genetically infeasible. Therefore, for the effective evolutionary emergence of complex behaviours, a modular developmental process is called for. For the sake of research validity (regarding long-term goals), this should be included from the outset.

Most artificial neural networks (ANNs) 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 behaviour-based work. Evidence from neuroscience provides further support, as biological neural networks are frequently recurrent. Although recurrent ANNs can be very hard to study (Boers and Kuiper 1992, p. 40), 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 organisms that are to act as complex dynamical systems working within a time frame.

6.1 Gruau's cellular encoding

Gruau (1996) used genetic programming techniques (Koza 1992) 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.

The crucial shortcoming is that modularity can only come from either gene duplication (see objections above) or iteration. But iteration is not a powerful enough 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 are variations between modules.

So, with GP techniques, 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.

6.2 Cellular automata

Many investigators have used cellular automata (CA) for the construction of neural networks, for example Gers and de Garis (1996) and Lee and Sim (1998). However, such work is more often at the level of neuron growth than the development of whole operational (rather than just large) networks. The working networks developed to date have been only basic. Although CA *rules* are suited to the evolution of network development in principle, the amount of work remaining makes this a major research hurdle.

6.3 Diffusion models

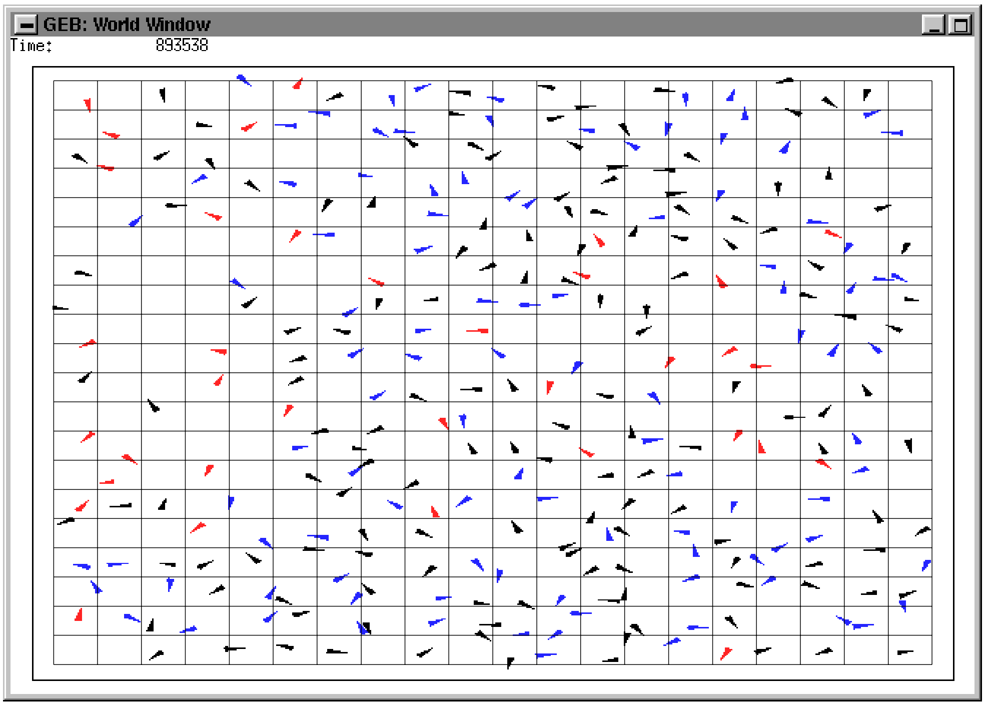
Although there are examples of work involving the evolution of neural networks whose development is determined by diffusion along concentration gradients, for example Vaario and Shimohara (1997), the resulting network structures have (to date) been only basic. So as to concentrate on the intended area of research, these models have also been passed over.

6.4 Lindenmayer systems

As mentioned above, developmental biology shows that genes provide a recipe for each cell to follow and that the activation of relevant genes is determined by a cell's immediate environment. All cells use the same set of rules, derived from the genes.

Lindenmayer systems (L-systems) were developed to model the biological growth of plants (Lindenmayer 1968). They are a class of fractals which apply *production rules* in parallel to the cells of their subject. 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 its immediate environment, including itself. In general, the most specific production rule that matches a cell's situation is applied.

Kitano (1990) used an L-system with context-free rules to evolve connectivity matrices. The number of rules in the genotype was variable. After each developmental step, the matrix 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 error back-propagation. He also showed that the number of rules could be small.



Time: 893538

Figure 2: The experimental world (Geb).

Boers and Kuiper (1992) used an L-system with context-sensitive rules to evolve modular feed-forward network architectures. A fixed-length alphabet was used for the rules, restricting the possible network architectures but still producing some good results. The evolution of production rules used a conventional genetic algorithm, with fixed-length genomes initially randomised. A limit of six rewrite passes over the network string was imposed.

Both these works used back-propagation to train the evolved networks. Also, the resulting structures were fully-connected clusters of unconnected nodes (i.e. no links within clusters and if one node in cluster A is linked to one node in cluster B then all nodes in A are linked to all nodes in B). It may be that the results achieved reflect the workings of back-propagation more than evolution. However, these works demonstrate the suitability of L-systems to non-iterative modular network development.

7 Experimental system definition

A system believed to be better suited to incremental artificial evolution by natural selection has been created, both to verify and extend the theory of evolutionary emergent systems generation presented thus far. Geb (named after the Egyptian god of the Earth) is a two-dimensional toroidal virtual world containing autonomous organisms, each controlled by a neural network. Neural networks were chosen with the aim of achieving sufficient genetic neutrality. Insufficient genetic neutrality is a pitfall in the evolution of computer program instructions, where no new genes (subroutines, at the pseudocode / descriptive level) have yet evolved. The networks are produced from bit-string genotypes by a developmental process, chosen with both genetic neutrality and the developmental requirements of the previous section in mind. No lifetime learning is used, to ensure that all results can be attributed to genetic evolution (an asset made clear in sections 5.2 and 6.4). Evolution within Geb is strictly by *natural selection*. There are no global system rules that delete organisms; this is under their own control.

Geb's world (figure 2) is divided into a grid of squares: 20 × 20 of them in most runs. No two individuals can 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. Individuals are otherwise free to move around the world, within and between squares. 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 direction.

Geb's main algorithm is detailed in algorithm 2.

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.

The steps involved for each individual are:

1. Network inputs are updated. See section 7.2.
2. Development – an iteration of the ontogenesis mechanism. See section 7.3.
3. All neural activations, including network outputs, are updated. See section 7.1.
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 7.2.

Algorithm 2: Geb's main algorithm.

7.1 Geb's neural networks

The ANNs used in Geb are recurrent networks of nodes as used successfully by Cliff, Harvey and Husbands in their evolutionary robotics work (Cliff, Harvey, and Husbands 1992; Harvey, Husbands, and Cliff 1992; Husbands, Harvey, and Cliff 1993). The neural model (figure 3) is based on McCulloch and Pitts' (1943) original proposal, which includes a distinct inhibitory mechanism (rather than the more prosaic positive-or-negative synaptic weights as typically used in parallel distributed processing systems). Cliff et al. evolved recurrent networks of these nodes for visual navigation tasks in simple environments.

The level of noise here (0.6 – see figure 3) 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* (Husbands, Harvey, and Cliff 1993), which are 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 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 unreasonably high. The neurons' veto threshold (0.5 – see function U in figure 3) is equal to the decision threshold for organisms' binary actions. All links have unit weight; no lifetime learning is used. This is to avoid the criticism that lifetime learning may be the main factor, as levelled at PolyWorld in section 5.2.

Each node has a bit-string *character* (label) attached to it, 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.

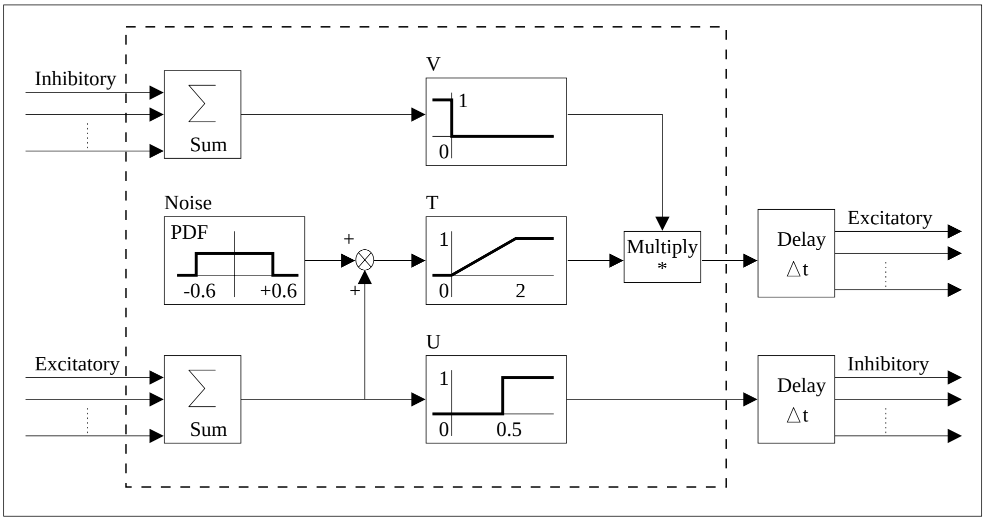


Figure 3: Schematic of a neuron, from (Cliff, Harvey, and Husbands 1992).

7.2 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 (if there is one)
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, bounded by unity as within any node. 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 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 not, the child replaces the individual being mated with. An organism cannot 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, always using 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-flip (bit-flip) on the child genotype. Figure 4 gives a simple illustrative example of the crossover and mutation used (although note that genotype lengths in the thousands would be more representative).

An organism's network input nodes have their excitatory inputs set to the weighted sum of the excitatory outputs from *matching* network output nodes' of other individuals in the neighbourhood.

```
Parent 1's genotype: 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 1
                                ^--- cut point in parent 1
Parent 2's genotype: 1 0 1 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1
                                  ^--- cut point in parent 2
------------------------------------------------------------
Child's    genotype: 1 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1
                       ^-- mutation point
```

Figure 4: Example of crossover and mutation.

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 characters 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 5×5 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.

7.3 Developmental system

A class of L-systems with context-free production rules 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' ANNs. A genotype determines the L-system's production rules which determine the organism's neural development. Thus, the production rules evolve.

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. So an empty (zero-length) predecessor matches any node's character and a predecessor cannot match a node's character that is shorter than it. The longer the matching predecessor, the better the match; the best matching rule (if any) is applied. Thus, ever more specific rules can evolve from those that have already been successful.

The production rules have the following form:

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

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) inherited 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 $\mathcal{S}_r$

The *successors* (1 and 2) are characters for the node(s) that replace the old node. 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. Necessary limits on the number of nodes and links are imposed.

The *replacement successor* (successor 1, if it has a character) is just the old (predecessor) node, with the same links but a different character. The *new successor* (successor 2, if it has a character) inherits a

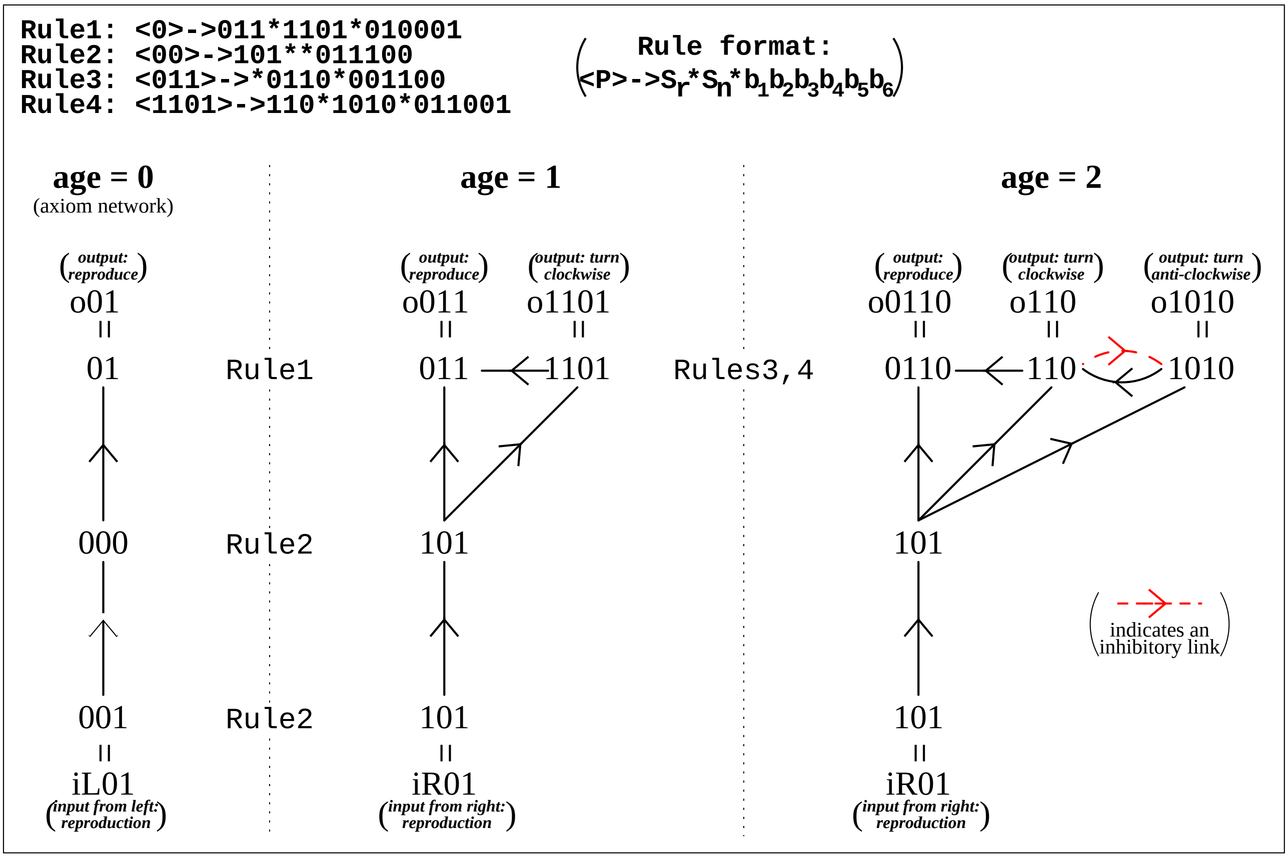


Figure 5: Example neural network development.

copy of the old node's input links unless it has a link from the old node (b_3 or b_4). 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).

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

The axiom network consists of three nodes with two excitatory links. The network output node's character (01) matches reproduction, the network input node's character (left input 01) matches this without matching any of the other action characters, and the hidden node's character neither matches nor is matched by the other nodes' or the action characters:

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

Development takes place throughout the individual's life, although necessary limits on the number of nodes and links are imposed. Figure 5 provides an example of the development of a very simple network.

7.4 Genetic decoding

The genetic decoding of production rules is loosely similar to that of Boers and Kuiper (1992). 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.

To read a rule, the system uses the concept 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

```
Genotype:  1 1 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 0
Rules:  i)  <>->   1 *   0 * 0 1 1 1 0 0
       ii)     <   1>->  0 *   1 * 1 0 0 0 0 0
Format:  < P >-> Sr *  Sn * link bits
```

Figure 6: Example rule generation.

start over, keeping track of the information bits of the segment. Note that a segment can be empty (have no information bits).

The full procedure to (attempt 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. If this is achieved before the end of the genotype then a rule is created. Figure 6 shows an example.

After reading all possible rules from a new-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, matching 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 memory required by Geb.

During this process, a further 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`. So, without this criterion, certain rules (such as those where P ends in `1`) would not be possible independently; they would automatically produce rules (such as $R{\rightarrow}N{,}B;C$) which interfere with their successors.

8 Results

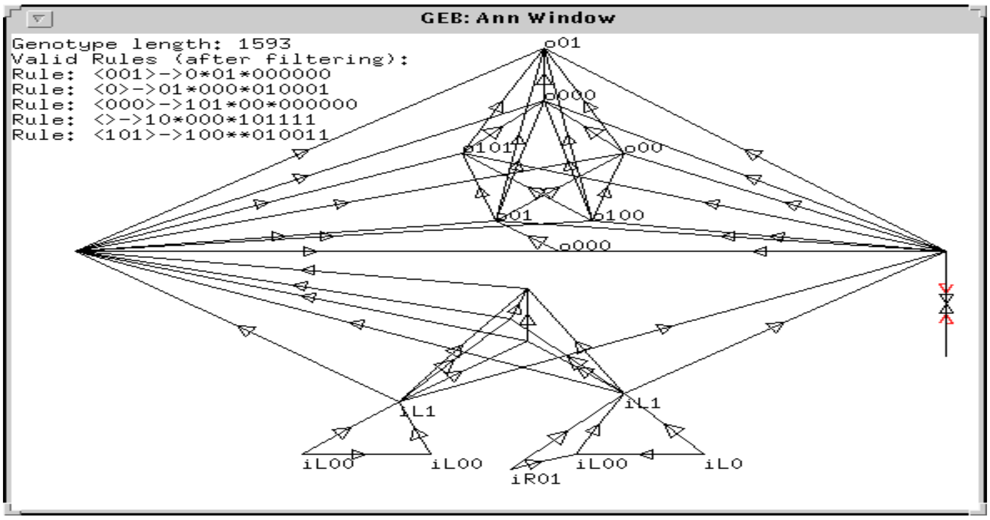
The system has consistently produced some important macro-level behaviours, although obviously the details of its evolution are different every time. This makes it difficult to describe the behaviour except by focusing on some typical and interesting observations.

8.1 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. 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. The criterion of a sufficiently correlated (implicit) fitness landscape has been met by the developmental system, making it suitable for long-term evolution. The remaining results are the proof of this suitability and so justify the claim that the use of neural networks can result in sufficiently correlated landscapes and further that Geb achieves this within a modular development system.

8.2 Emergent collective behaviour

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



```
Genotype length: 1593
Valid Rules (after filtering):
Rule: <001>->0*01*000000
Rule: <0>->01*000*010001
Rule: <000>->101*00*000000
Rule: <>->10*000*101111
Rule: <101>->100**010011
```

Figure 7: A dominant organism's neural network.

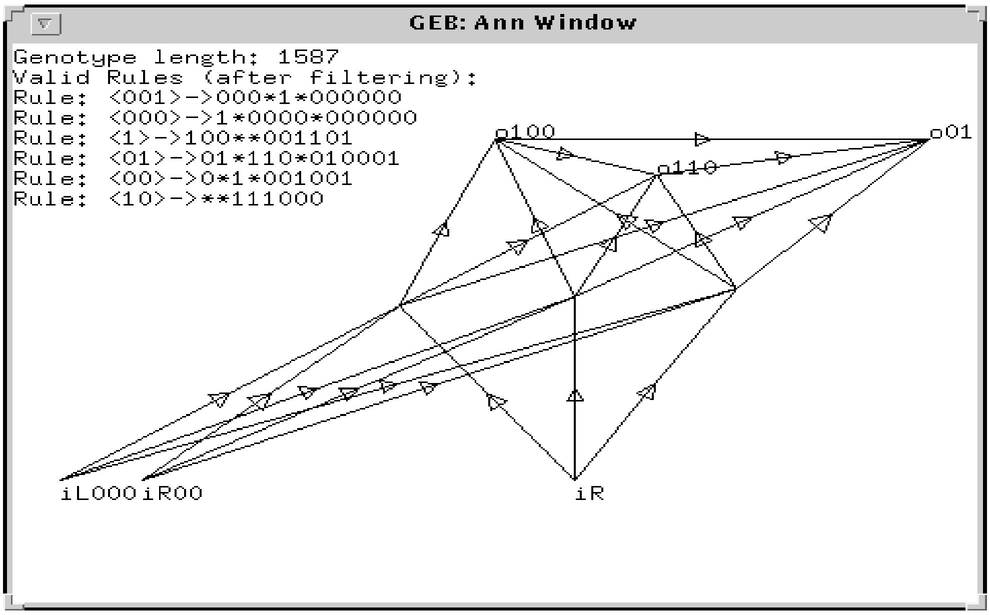
population. Some networks do better than others but not sufficiently well for them to display a dominating effect on Geb's world window.

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 7 shows an example of such an individual. Network outputs are prefixed with o, inputs with i. Input characters are shown with their first bit translated from 0,1 to L,R (left,right). 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 it clear why this is so advantageous. The individuals keep each other moving quickly, in tight circles. Any attacking organism would have to either get its timing exactly right or approach in a faster spiral – both relatively advanced strategies. These dominant individuals also mate just before killing. The offspring (normally) appear beyond the individual being killed, away from the killer's path.

8.3 Naturally arising coevolution

Because of the success of this first dominant species (especially their success at killing other organisms), the world always has enough space for other organisms to exist. Such other organisms tend not to last long; almost any movement will bring them into contact with one of the dominant organisms, helping that species in its reproduction as much as themselves. However, they can make some progress. Individuals have emerged that are successful at turning to face members of 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 7) is shown in figure 8. Note that most rebels have many more nodes and links; this one was picked for its clarity. The main points to note from this figure are the network inputs iL000 (left 000) and iR00 (right 00) which match the very non-action output characters that members of the dominant species use to support each other's activations (o000 [x2], o00). The network outputs are o100 (fight), o01 (reproduce) and o110 (turn clockwise). By turning clockwise, a rebel will turn towards its enemy fastest when the enemy is to its right, which is where most of the rebel's input is taken from (via the input iR), and hence the side it is best at responding to. Most rebels have more complicated networks, which are very difficult to comprehend. Indeed, many of these have proved unassailably difficult to understand in detail.



```
Genotype length: 1587
Valid Rules (after filtering):
Rule: <001>->000*1*000000
Rule: <000>->1*0000*000000
Rule: <1>->100**001101
Rule: <01>->01*110*010001
Rule: <00>->0*1*001001
Rule: <10>->**111000
```

Figure 8: A rebel organism's neural network.

8.4 Ongoing coevolution

Figures 9 and 10 show *running averages* of the number of organisms reproducing and killing, from two typical experimental runs. Each point is the average of the raw data (number of appropriate organisms) over a window moving along the time axis. This filters out Lotka-Volterra population cycles and short term random variations, revealing long term shifts. These figures suggest that further species emerge, indicating ongoing evolutionary emergence. However, organisms have proved difficult to analyse beyond the above, even at the behavioural level. All that can currently be said is that they share characteristics of the previous species but are different.

While it was expected that inhibitory links would play an important role, 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.

9 Conclusions

The emergence of increasingly complex advantageous behaviours requires the perpetuation of evolutionary emergence. While computational emergence can arise via artificial selection, evolutionary emergence requires natural selection (by our definitions). The logical progression or aim is the perpetuation of evolutionary emergence via naturally arising coevolution. However, this requires long-term incremental evolution and so what we evolve and how we evolve it must be chosen accordingly. The initial groundwork on "how" has already been covered by SAGA theory – by using low enough mutation rates that the population evolves as nearly-converged species, with crossover assimilating beneficial mutations into the species. As for what class of entities to (attempt to) evolve, computer program instructions are too brittle. Even the use of template matching cannot overcome that fact. Neural networks are a clear choice because of their graceful degradation.

Natural selection research should be leading the way, through the evolution of neural controllers within virtual environments, towards the emergence of increasingly complex advantageous behaviours. The work presented in this paper has started down that route, with some success. In work involving pure natural selection, the organisms' developmental and interaction systems are analogous to the fitness functions of conventional genetic algorithms. While the general aim involves moving away from such comparisons, the analogy is useful for recognising how the epistasis (lack of correlation) of fitness landscape issue transfers across. Certain ontogenetic (developmental) and

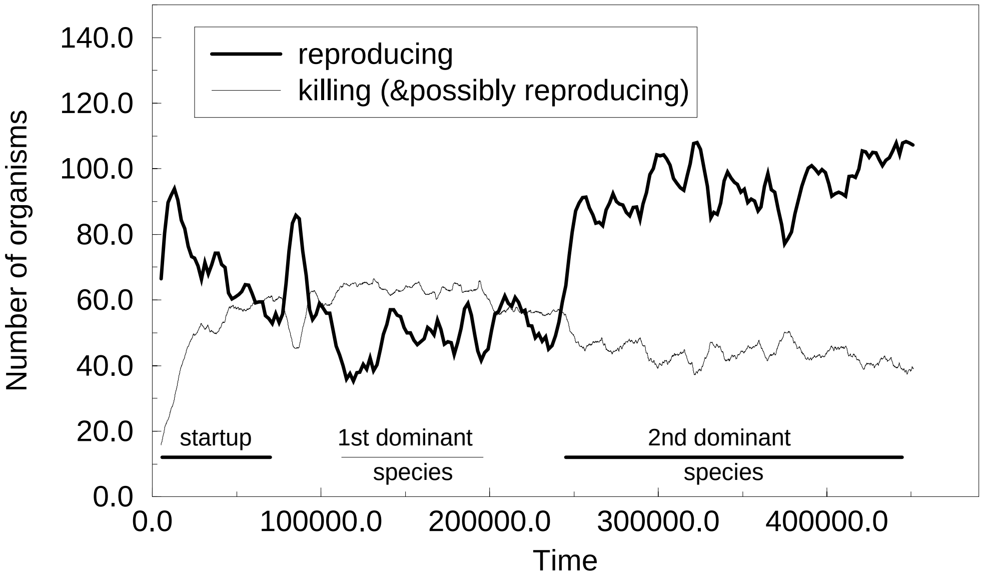


Figure 9: Typical run 1 (running averages of population sizes by actions).

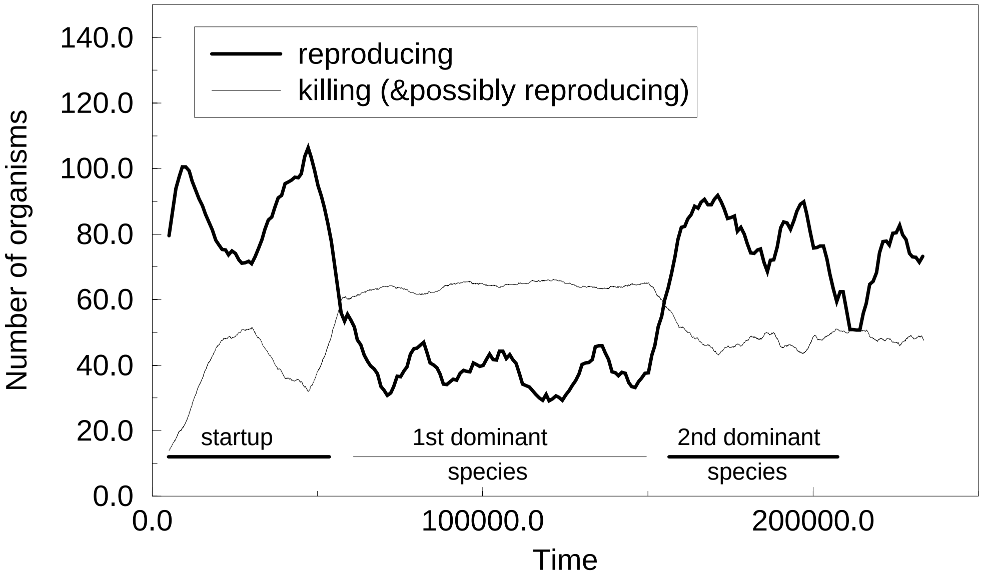


Figure 10: Typical run 2 (running averages of population sizes by actions).

interaction systems can result in individuals with similar genotypes but very different phenotypes. Geb organisms satisfy this criterion, because offspring resemble their parents (but are not identical). Geb's results prove it to be suited to long-term incremental artificial evolution. This alone is a significant result for a modular developmental system. The behaviours identified are encouraging too, for the increases in complexity were clearly advantageous and in ways not specified by the design – evolutionary emergence.

Geb provides some useful lessons for the development of the field towards the perpetuation of such emergence. First the transparency of behaviours has surfaced as an important issue. Neural networks and other such highly distributed controllers can be suited to long-term evolution, but analysis of evolved networks soon becomes infeasible as their complexity increases. This was a less significant problem in the evolution of program code. The recommendation is, therefore, that future systems should be developed such that behavioural descriptions are as easy to generate as possible, probably by constructing the systems such that behaviours will be transparent to human observers.

Another lesson learned concerns the specification of lowest-level actions. Geb generated the important new result of evolutionary emergent advantageous behaviours from a system suited to long-term incremental evolution. However, alternatives in which the evolvable embodiment of an organism gives rise to its actions will be necessary for the open-ended evolution of available actions. This could provide for a far greater range of behaviours, no longer restricted to sequences of predefined actions.

Future work on the current system includes testing the sensitivity of results to variations in parameters such as grid size, maximum population size (number of grid squares), mutation rate and level of noise in the networks. However, although improvements may be possible by tuning parameters, greater advances will probably be produced by heeding the above lessons.

This work holds interest not just for those within the artificial evolution field, but for anyone interested in the generation of systems which outperform their design specifications (cf. Descartes' Dictum). For there is probably no process other than natural selection that is capable of producing the open-ended emergence of increasingly complex systems.

Acknowledgements

This work is supported by an award from the United Kingdom's Engineering and Physical Sciences Research Council to author ADC. It is a continuation of previous work (Channon 1996) also supported by an award from the EPSRC (supervisor Inman Harvey, University of Sussex).

References

Adami, C. and C. T. Brown (1994). Evolutionary learning in the 2D artificial life system 'Avida'. In R. A. Brooks and P. Maes (Eds.), *Proceedings of Artificial Life IV*, pp. 377–381. Cambridge, MA: MIT Press.

Boers, E. J. W. and H. Kuiper (1992). Biological metaphors and the design of modular artificial neural networks. Master's thesis, Departments of Computer Science and Experimental Psychology, Leiden University, The Netherlands.

Brooks, R. A. (1991a). Intelligence without reason. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pp. 569–595. San Mateo, CA: Morgan Kauffman.

Brooks, R. A. (1991b). Intelligence without representation. *Artificial Intelligence 47*, 139–159.

Bullock, S. G. (1995). Co-evolutionary design: Implications for evolutionary robotics. Technical Report CSRP384, University of Sussex School of Cognitive and Computing Sciences.

Cariani, P. (1991). Emergence and artificial life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (Eds.), *Proceedings of Artificial Life II*, pp. 775–797. Redwood City, CA: Addison-Wesley.

Channon, A. D. (1996). The evolutionary emergence route to artificial intelligence. Master's thesis, School of Cognitive and Computing Sciences, University of Sussex.

Channon, A. D. and R. I. Damper (1998a). Evolving novel behaviors via natural selection. In C. Adami, R. Belew, H. Kitano, and C. Taylor (Eds.), *Proceedings of Artificial Life VI, Los Angeles*, pp. 384–388. Cambridge, MA: MIT Press.

Channon, A. D. and R. I. Damper (1998b). Perpetuating evolutionary emergence. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. Wilson (Eds.), *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB98), Zurich*, pp. 534–539. Cambridge, MA: MIT Press.

Cliff, D., I. Harvey, and P. Husbands (1992). Incremental evolution of neural network architectures for adaptive behaviour. Technical Report CSRP256, University of Sussex School of Cognitive and Computing Sciences.

Cliff, D. and G. Miller (1995). Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón (Eds.), *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, pp. 200–218. New York: Springer Verlag.

Dawkins, R. and J. R. Krebs (1979). Arms races between and within species. *Proceedings of the Royal Society of London B 205*, 489–511.

Fogel, L. J. (1962). Autonomous automata. *Industrial Research 4*, 14–19.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution*. New York: John Wiley.

Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Sciences 10*, 484–491.

Gers, F. and H. de Garis (1996). CAM-Brain: A new model for ATR's cellular automata based artificial brain project. In *Proceedings of ICES'96: International Conference on Evolvable Systems*.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Gruau, F. (1996). Artificial cellular development in optimization and compilation. Technical report, Psychology Department, Stanford University, Palo Alto, CA.

Harnad, S. (1993). Artificial life: Synthetic vs. virtual. In C. G. Langton (Ed.), *Proceedings of Artificial Life III*, pp. 539–552. Reading, MA: Addison-Wesley.

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

Harvey, I. (1993a). *The Artificial Evolution of Adaptive Behaviour*. Ph. D. thesis, School of Cognitive and Computing Sciences, University of Sussex.

Harvey, I. (1993b). Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In C. G. Langton (Ed.), *Proceedings of Artificial Life III*, pp. 299–326. Reading, MA: Addison-Wesley. Also available as report CSRP222, School of Cognitive and Computing Sciences, University of Sussex.

Harvey, I. (1997). Cognition is not computation: Evolution is not optimisation. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (Eds.), *Proceedings of International Conference on Artificial Neural Networks (ICANN 97), Special Session on Adaptive Autonomous Agents at ICANN97, Lausanne, Switzerland*, pp. 685–690. Berlin: Springer-Verlag.

Harvey, I., P. Husbands, and D. Cliff (1992). Issues in evolutionary robotics. In J. A. Meyer, H. Roitblat, and S. Wilson (Eds.), *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92)*, pp. 364–373. Cambridge, MA: MIT Press / Bradford Books. Also available as report CSRP219, School of Cognitive and Computing Sciences, University of Sussex.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D 42*, 228–234.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM 9*(3), 297–314.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press. 2nd edition: MIT Press, 1992.

Holland, J. H. (1992). Genetic algorithms. *Scientific American 267*(1), 44–50.

Husbands, P., I. Harvey, and D. Cliff (1993). Analysing recurrent dynamical networks evolved for robot control. In *Proceedings of the Third IEE International Conference on Artificial Neural Networks (IEE-ANN93)*, pp. 158–162. London: IEE Press.

Kauffman, S. and S. Levin (1987). Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology 128*, 11–45.

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

Koza, J. R. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Department of Computer Science, Stanford University, Palo Alto, CA.

Koza, J. R. (1992). *Genetic Programming*. Cambridge, MA: MIT Press / Bradford Books.

Koza, J. R. (1993). Artificial life: Spontaneous emergence of self-replicating and evolutionary self-improving computer programs. In C. G. Langton (Ed.), *Proceedings of Artificial Life III*, pp. 225–262. Reading, MA: Addison-Wesley.

Langton, C. G. (1989). Artificial life. In C. G. Langton (Ed.), *Proceedings of Artificial Life*, pp. 1–47. Redwood City, CA: Addison-Wesley.

Lee, D. W. and K. B. Sim (1998). Evolving cellular automata neural systems 1 (ECANS1). In *Proceedings of The Third Asian Fuzzy System Symposium, Masan, Korea*, pp. 158–163.

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development. *Journal of Theoretical Biology 18*, 280–315. Parts I and II.

Lotka, A. J. (1925). *Elements of Physical Biology*. Baltimore, OH: Williams and Wilkins. Reprinted as *Elements of Mathematical Biology*, Dover Press, 1956.

Mayr, E. (1960). The emergence of evolutionary novelties. In S. Tax (Ed.), *Evolution after Darwin, The University of Chicago Centennial, Vol. I: The Evolution of Life*, pp. 349–380. Chicago, IL: University of Chicago Press.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5*, 115–133.

Packard, N. H. (1989). Intrinsic adaptation in a simple model for evolution. In C. G. Langton (Ed.), *Proceedings of Artificial Life*, pp. 141–155. Redwood City, CA: Addison-Wesley.

Ray, T. S. (1991). An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (Eds.), *Proceedings of Artificial Life II*, pp. 371–408. Redwood City, CA: Addison-Wesley.

Ray, T. S. (1992). Evolution, ecology and optimization of digital organisms. Technical Report 92-08-042, Santa Fe Institute, Santa Fe, NM.

Ray, T. S. (1996). Artificial life. In W. Gilbert and G. T. Valentini (Eds.), *From Atoms to Minds*. Rome: Istituto della Enciclopedia Italiana Treccani.

Ray, T. S. (1997). Selecting naturally for differentiation. In J. R. Koza, K. D. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pp. 414–419. San Francisco, CA: Morgan Kaufmann.

Reynolds, C. W. (1992). An evolved, vision-based behavioral model of coordinated group motion. In J. A. Meyer, H. Roitblat, and S. Wilson (Eds.), *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92),* pp. 384–392. Cambridge, MA: MIT Press.

Reynolds, C. W. (1994). Competition, coevolution and the game of Tag. In R. A. Brooks and P. Maes (Eds.), *Proceedings of Artificial Life IV*, pp. 59–69. Cambridge, MA: MIT Press.

Rucker, R. (1993). *Artificial Life Lab.* Corte Madera, CA: Waite Group Press.

Sannier, A. V. and E. D. Goodman (1987). Genetic learning procedures in distributed environments. Technical report, A.H. Case Center for Computer-Aided Engineering and Manufacturing, Michigan State University, East Lansing, MI.

Skipper, J. (1992). The computer zoo – evolution in a box. In F. J. Varela and P. Bourgine (Eds.), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life,* pp. 355–364. Cambridge, MA: MIT Press / Bradford Books.

Steels, L. (1994). The artificial life roots of artificial intelligence. *Artificial Life Journal 1*(1), 89–125.

Stephan, A. (1998). Varieties of emergence in artificial and natural systems. *Zeitschrift fur Naturforschung C-A Journal of Biosciences 53*(7–8), 639–656.

Stork, D. G., B. Jackson, and S. Walker (1991). ‘Non-optimality’ via pre-adaptation in simple neural systems. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (Eds.), *Proceedings of Artificial Life II*, pp. 409–429. Redwood City, CA: Addison-Wesley.

Taylor, T. and J. Hallam (1997). Studying evolution with self-replicating computer programs. In P. Husbands and I. Harvey (Eds.), *Proceedings of the Fourth European Conference on Artificial Life (ECAL97), Brighton*, pp. 550–559. Cambridge, MA: MIT Press.

Vaario, J. and K. Shimohara (1997). Synthesis of developmental and evolutionary modeling of adaptive autonomous agents. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (Eds.), *Proceedings of International Conference on Artificial Neural Networks (ICANN 97), Special Session on Adaptive Autonomous Agents at ICANN97, Lausanne, Switzerland*, pp. 721–725. Berlin: Springer-Verlag.

Volterra, V. (1926). Variations and fluctuations of the number of individuals in animal species living together. In R. N. Chapman (Ed.), *Animal Ecology*, pp. 409–448. New York: McGraw-Hill.

Wilson, S. W. (1991). The animat path to AI. In S. W. Wilson and J.-A. Meyer (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB),* pp. 15–21. Cambridge, MA: Bradford Books / MIT Press.

Yaeger, L. (1993). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or PolyWorld: Life in a new context. In C. G. Langton (Ed.), *Proceedings of Artificial Life III*, pp. 263–298. Reading, MA: Addison-Wesley.

Zaera, N., D. Cliff, and J. Bruten (1996). (Not) evolving collective behaviours in synthetic fish. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. Wilson (Eds.), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96)*, pp. 635–644. Cambridge, MA: MIT Press / Bradford Books.