Perpetuating evolutionary emergence

A.D. Channon and R.I. Damper
Image, Speech & Intelligent Systems Research Group
University of Southampton, Southampton, SO17 1BJ, UK
http://www.soton.ac.uk/~adc96r
adc96r@soton.ac.uk rid@ecs.soton.ac.uk

Abstract

Perpetuating evolutionary emergence is the key to artificially evolving increasingly complex systems. In order to generate complex entities with adaptive behaviors beyond our manual design capability, long-term incremental evolution with continuing emergence is called for. Purely artificial selection models, such as traditional genetic algorithms, are argued to be fundamentally inadequate for this calling and existing natural selection systems are evaluated. Thus some requirements for perpetuating evolutionary emergence are revealed. A new environment containing simple virtual autonomous organisms has been created to satisfy these requirements. Resulting evolutionary emergent behaviors are reported alongside of their neural correlates. In one example, the collective behavior of one species clearly provides a selective force which is overcome by another species, demonstrating the perpetuation of evolutionary emergence via naturally arising coevolution.

1. Evolutionary emergence

Emergence is related to qualitatively novel structures and behaviors which are not reducible to those hierarchically below them. It poses an attractive methodology for tackling Descartes' Dictum: "how can a designer build a device which outperforms the designer's specifications?" (Cariani, 1991, page 776). Most importantly, it is necessary for the generation of complex entities with behaviors beyond our manual design capability.

Cariani identified the three current tracts of thought on emergence, calling them "computational", "thermodynamic" and "relative to a model" (Cariani, 1991). Computational emergence is related to the manifestation of new global forms, such as flocking behavior 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 in order to keep up with a system's behavior. This is close to Steels' 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" (Steels, 1994, section 4.1).

Evolutionary emergence falls into the 'emergence relative to a model' category. Consider a virtual world of organisms that can move, reproduce and kill according to rules sensitive to the presence of other organisms, evolving under natural selection. Should flocking manifest itself in this system, we could classify it as emergent in two senses: firstly in the 'computational' sense from the interaction of local rules, flocking being a collective behavior, and secondly in the 'relative to a model' sense from the evolution, the behavior being novel to the system. While the first is also relevant to our goal, in that complex adaptive systems will involve such emergence, the second is the key to understanding *evolutionary* emergence.

Harvey's Species Adaptation Genetic Algorithm (SAGA) theory (Harvey, 1992) provides a framework for incremental evolution, necessary for evolutionary emergence. In this paradigm a population, with possibly just a few tens of members, evolves for many thousands of generations, with gradual changes in genotype information content. Increases in complexity must therefore result from evolution itself. This is in contrast to the common use of the Genetic Programming (GP) paradigm, 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 programs result in rugged fitness landscapes, there can be little further evolution of this converged population. Here we see one of the requirements of SAGA: a smooth fitness landscape.

Having specified what is meant by evolutionary emergence, we will now explore the two types of selection which might be used to bring evolutionary emergence about. Packard 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" (Packard, 1989, abstract). We will refer to them as 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.

2. Artificial selection

Within the artificial evolution field, variants of the optimization paradigm have proven fruitful. Even where the concepts of SAGA theory are dominant, practice still holds to the use of fitness functions. But as the complexity of behaviors attempted increases, flaws in the artificial selection approach are appearing. Zaera, Cliff and Bruten's failed attempts at evolving schooling behavior in artificial 'fish' (Zaera et al., 1996) provide an account of the difficulties faced. An extract from the abstract of their paper still yields an excellent summary of the state of artificial selection work within the field:

> "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 (Zaera et al., 1996, section 5). They cited two promising works: Reynolds' evolution of coordinated group motion in 'prey' animats pursued by a hard-wired 'predator' (Reynolds, 1992), and Rucker's 'ecosystem' model (Rucker, 1993) 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 dominant approach in the adaptive behavior field. This is essentially a response to the problems encountered when trying to use artificial selection to evolve complex behaviors. However, artificial selection has kept its hold so far – most systems still use fitness functions. The reasoning given for imposing coevolution is often that it helps in overcoming problems arising from the use of static fitness landscapes.

From the discussion so far, one might assume our argument to be 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 neither sufficient nor necessary. *In the context of evolutionary emergence, any artificial selection used constitutes just one of the parts of a system.* Artificial selection can only select for that which it is specified to. Therefore anything that emerges during evolution *must* be due to another aspect of selection, which must in turn be due to the innate dynamics of the system – natural selection.

3. Natural selection

As noted in section 1, genetic codings of computer programs result in rugged fitness landscapes, making them unsuitable for incremental evolution. However, most natural selection work has been program code evolution, following the initial success of 'Tierra' (Ray, 1991).

3.1 Natural selection of program code

Tierra is a system of self-replicating machine code programs, initialized as a single manually designed self-replicating program. To make evolution possible, random bit-flipping was imposed on the memory. A degree of artificial selection was imposed by *the system* deleting the oldest programs in order to free memory, with an added bias against programs that generated error conditions.

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 (such as 13). Thus direct memory addressing was not possible. Instead, the manually designed program used consecutive NOP (No-OPeration) instructions which acted as templates that could be found by certain machine code instructions. This 'addressing by templates' is how the program determined the points at which to begin and end copying. Another aspect of the system was that computational errors were introduced at random. 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; they replicated by executing the copy loop from neighboring 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. Ray reported that coevolution occurred as the hosts became immune to the parasites, which overcame these defenses, 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 parasites. 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. Another class of adaptations found was "optimizations". 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, there have been no new reports of emergent phenomena during the last few years. It is generally accepted that not much more will occur unless further alterations are made to the system.

The evolvability of the code would seem 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 following pseudo-code of the initial, manually designed program:

```
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  } procedure
      a++ ; b++ ; jump to T1010        }
T1011 pop c,b,a off stack ; return     }
T1110
```

This pseudo-code is based on the initial program as listed in (Ray, 1992, appendix C). The T????s on the left denote four-bit templates. Now, as reported in (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 procedure. Further, a 'host' that is immune to such a parasite can also be produced by a single-bit mutation in the template-comparison 1011 in the "if c=0 then jump to T1011" statement of the copy procedure; 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, it is clear that these phenomena are due to the flexibility of the template matching. It is also clear that the same argument applies to the other ecological adaptations reported.

While this demonstrates the flexibility of template matching at the four-bit level, it also shows the ecological results to be somewhat more trivial than we might have first hoped. For we must hold to our previous argument that programs are not suitable for incremental evolution; complex programs would necessarily contain many more templates, which would have to be significantly longer. Thus we would pass far beyond the trivial four-bit templates that random search can operate on, to a stage where evolution is impractical.

There is still the issue of the 'optimization' adaptations to address. Two of the results, 'unrolled' copy loops and efficient replication by parallel-processing, only needed evolution to insert (probably by the action of the random computational errors) repetitions of neighboring 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 (Ray, 1992, appendix C) shows that the transition can be made by simply deleting instructions (mostly NOPs – cutting out the redundancy in the templates) and just six mutations, three of which are unnecessary. So apart from three mutations, this is a 'less of the unnecessary' solution – certainly not sufficient to challenge our argument.

3.2 Natural selection of more suitable entities

Although the approach of most natural selection work to date has been to evolve program code, there are two notable exceptions. The second (chronologically) is Channon's own work, which was conceived independently of and initially created in ignorance of both Tierra (including its derivatives) and the first exception: "PolyWorld" (Yaeger, 1993). PolyWorld simulates many aspects of the real world including energy conservation (an implicit fitness function), food, movement (on a 2D plane), vision, neural networks and learning.

PolyWorld organisms have seven pre-programmed behaviors: eating, mating, fighting, moving, turning, focusing and lighting. 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, including 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. Yaeger reported a variety of emergent behaviors, including 'fleeing', 'fighting back', 'grazing', 'foraging' and 'following'.

One criticism of PolyWorld, in the context of perpetuating evolutionary emergence, is that (Hebbian) learning may well be overwhelmingly responsible for the results. There is little evidence of significant evolution at the genetic level; 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, 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. While this is valid evolutionary emergence, it is not sufficient for perpetuating evolutionary emergence. Unless learning is evolvable or what is learnt can be passed on, a maximal level will be reached at which organisms are not capable of learning more.

4. A new environment – model definition

A system believed to be better suited to incremental artificial evolution by natural selection has been created. '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. The killing of organisms is under their own control.

Geb's world is divided into a grid of squares; 20 × 20 of them in most runs. No two individuals may occupy the same square at any one time. This gives the organisms a 'size' and puts a limit on their number. They are otherwise free to move within and between squares.

Initialization: Individuals have single-bit genotypes '0'.
Main Loop: For each individual:

1. Update network inputs.
2. Development – one iteration.
3. Update all neural activations, including network outputs.
4. Carry out actions associated with network outputs.

4.1 The neural networks

The neural networks used in Geb are recurrent networks of nodes as used successfully by Cliff, Harvey and Husbands in their evolutionary robotics work (figure 1).

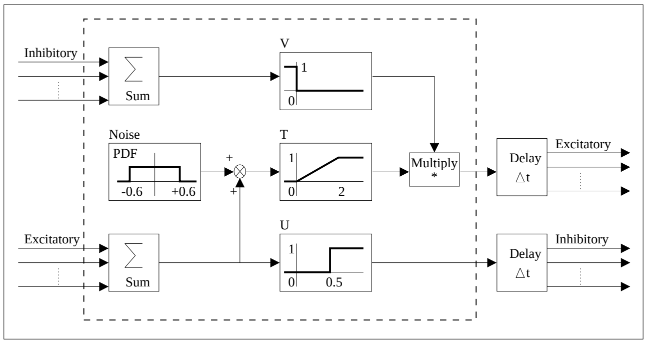

Figure 1 Schematic of a neuron, from (Cliff et al., 1992)

All links have unit weight; no lifetime learning is used. This is to avoid criticism that lifetime learning may be the main factor, as levelled at PolyWorld in section 3.2. Each node has a bit-string 'character' (label) attached to it. This is used to match organisms' inputs, outputs and actions, and to determine the node's development.

4.2 Organism $\longleftrightarrow$ 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, the organism will turn clockwise by an angle proportional to the node's excitatory output. If an action has more than one matching output node then the relevant output is the sum of these nodes' excitatory outputs, bounded by unity as within any node. If an action has no matching output node, then the relevant output is noise, at the same level as in the (other) nodes.

An organism's input nodes have their excitatory inputs set to the weighted sum of 'matching' output nodes' excitatory outputs from other individuals in the neighborhood. If the first bit of an input node's character is 1 then its input is from individuals to the right, otherwise from individuals to the left. An input node 'matches' an output node if the rest of its character is the same as the start of the output node's character. Weighting is inversely proportional to distance between individuals.

When an organism reproduces with a mate in front of it, the child is placed in the square beyond the mate if that square is empty. If it is not then the child replaces the mate unless the mate is fighting, in which case the child is killed. Reproduction involves crossover, with the cut point always offset by one gene (either way), and a single gene-flip (bit-flip) mutation.

4.3 The developmental system

A context-free L-system was designed in which children's networks resemble aspects of their parents'. The axiom network consists of three nodes with two excitatory links:

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

The production rules have the following form:

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

$\mathcal{P}$	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 $\mathcal{S}_r$

For each node, the production rule with the longest matching predecessor is applied. Thus ever more specific rules can evolve. If a successor has no character then that node is not created. A 'replacement' successor is just the old (predecessor) node with its character changed. A 'new' successor inherits a 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 network input nodes are (only) produced from network input nodes and new output nodes (only) from output nodes. The character-based method of matching up network inputs and outputs ensures that the addition or removal of an input/ output node at a later stage of development or evolution will not damage the relationships of previously adapted inputs and outputs.

Details of the theory behind the choice of developmental system, and of the genetic decoding of production rules, can be found in (Channon and Damper, 1998).

5. Results

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 the 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.

5.1 Emergent collective behavior

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.

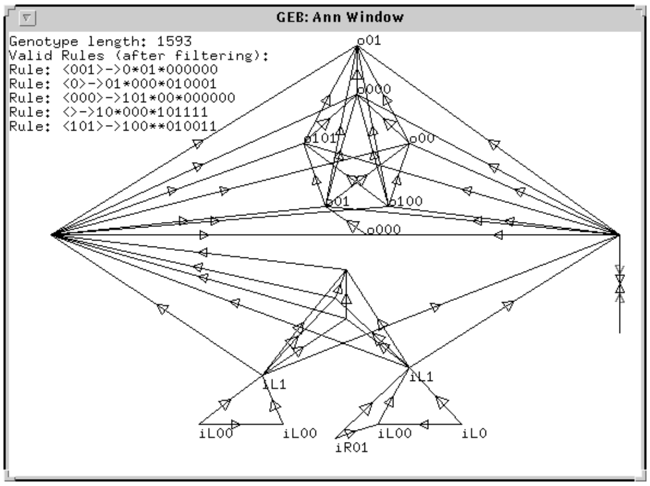

Figure 2 A Dominant Organism

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 2 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. 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 individuals also mate just before killing. The offspring (normally) appear beyond the individual being killed, away from the killer's path.

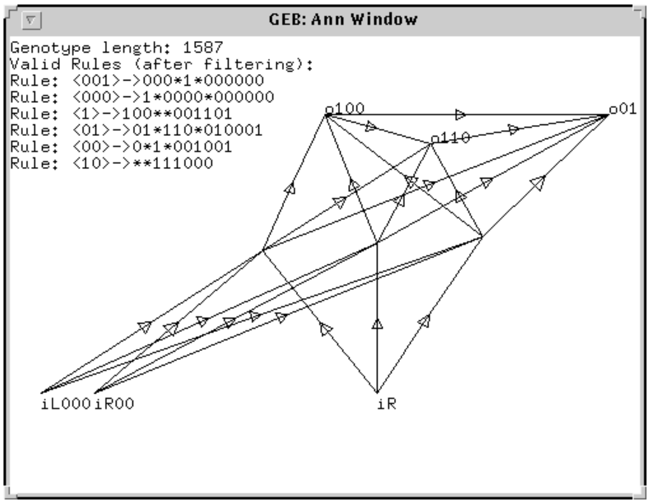

Figure 3 A rebel

5.2 Naturally arising coevolution

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 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 2) is shown in figure 3. Note that most rebels have many more nodes and links; this one was picked for its clarity.

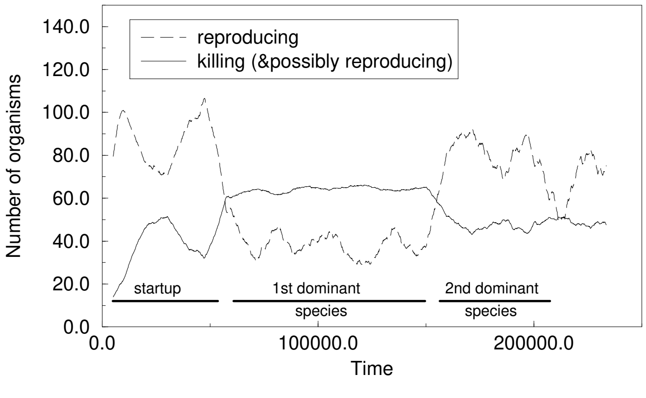

Figure 4 Typical run (running averages)

Further, running averages of the number of organisms reproducing and killing (figure 4) suggest that further species emerge, indicating perpetuating evolutionary emergence. However, organisms have proved difficult to analyze beyond the above, even at the behavioral level. All that can currently be said is that they share characteristics of the previous species but are different.

6. Conclusions

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. As for what to evolve, program code is too brittle. Even the use of template matching cannot overcome that fact. Neural networks are a clear choice because of their graceful degradation (high degree of neutrality).

The implication for animat research is that it should be leading the way, through the natural selection of neural controllers, towards the emergence of ever more complex and impressive behaviors. The work presented in this paper has started down that route, with some success. Geb organisms satisfy the convergence criterion, because offspring resemble their parents (but are not identical). So Geb is suited to long-term incremental artificial evolution. The behaviors identified are encouraging too, for the increases in complexity were in ways not specified by the design – evolutionary emergence. These are the two cardinal quality targets for any such work.

Whether on not emergence is continuing in Geb is hard to tell, for it soon becomes difficult to identify behaviors. This was a less significant problem in the evolution of program code but evolved neural networks are hard to understand and so offer little help. Constructing systems such that behaviors will be more transparent is likely to be the most productive way forward.

A further problem is that specifying the available 'actions' constrains organisms around these actions and so limits evolution. Despite showing the important new result of evolutionary emergent behaviors (not specified within the initial system) from a system suited to long-term incremental evolution, all basic (inter-)actions were as specified within the initial system and not evolvable. At a later stage, alternatives in which the evolvable embodiment of an organism gives rise to its actions will need to be considered. Only then will we have a truly open stage on which to watch the perpetuation of evolutionary emergence.

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

Cariani, P. (1991). Emergence and artificial life. In *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity, Vol. X*, pages 775–797, Redwood City, CA.

Channon, A. (1996). The evolutionary emergence route to artificial intelligence. Master's thesis, School of Cognitive and Computing Sciences, University of Sussex. Revision October 1996. http://www.soton.ac.uk/~adc96r/.

Channon, A. and Damper, R. (1998). Evolving novel behaviors via natural selection. In Adami, C., Belew, R., Kitano, H., and Taylor, C., editors, *Proceedings of "Artificial Life VI", Los Angeles, June 26-29, 1998.* MIT Press. http://www.soton.ac.uk/~adc96r/.

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

Harvey, I. (1992). Species adaptation genetic algorithms: A basis for a continuing SAGA. In Varela, F. and Bourgine, P., editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354, Cambridge, MA. MIT Press/ Bradford Books.

Harvey, I. (1997). Cognition is not computation: Evolution is not optimisation. In *forthcoming Proceedings of ICANN97. Special Session on Adaptive Autonomous Agents at ICANN97.*

Packard, N. H. (1989). Intrinsic adaptation in a simple model for evolution. In Langton, C., editor, *Artificial Life, Santa Fe Institute Studies in the Sciences of Complexity, Vol. VI*, pages 141–155. Addison-Wesley.

Ray, T. S. (1991). An approach to the synthesis of life. In Langton, C., Taylor, C., Farmer, J., and Rasmussen, S., editors, *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity, Vol. X*, pages 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.

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

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

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

Yaeger, L. (1993). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or polyworld: Life in a new context. In Langton, C. G., editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Vol. XVII*, pages 263–298.

Zaera, N., Cliff, D., and Bruten, J. (1996). (Not) evolving collective behaviours in synthetic fish. In Maes, P., Mataric, M., Meyer, J.-A., Pollack, J., and Wilson, S., 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.