Unbounded Evolutionary Dynamics in a System of Agents that Actively Process and Transform their Environment [*]

Alastair Channon (alastair@channon.net)
School of Computer Science, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom

Abstract. Bedau et al.'s statistical classification system for long-term evolutionary dynamics provides a test for open-ended evolution. Making this test more rigorous, and passing it, are two of the most important open problems for research into systems of agents that actively process and transform their environment. This paper presents a detailed description of the application of this test to 'Geb', a system designed to verify and extend theories behind the generation of evolutionarily emergent systems. The result is that, according to these statistics, Geb exhibits unbounded evolutionary dynamics, making it the first autonomous artificial system to pass this test. However, having passed it, the most prudent course of action is to look for weaknesses in the test. The test is criticized, most significantly with regard to its normalization method for artificial systems. Furthermore, this paper presents a modified normalization method, based on component activity normalization, that overcomes these criticisms. The results of the revised test, when applied to Geb, indicate that this system does indeed exhibit open-ended evolution.

Keywords: evolutionary dynamics, variable-size genomes, coevolution, biotic selection, emergence

1. Introduction

As articulated in the aims of this journal, the challenge for researchers in this field is to mirror the process of neo-Darwinian evolution so as to enable the automatic evolution of components that actively process environmental information and transform their environment. Such automaticity takes us beyond the ability of evolutionary algorithms to find optimal solutions in static environments, into issues of evolvability in dynamic environments, and even on toward unbounded (or 'open-ended') evolution. In order to determine our progress in this venture, we need quantifiable measures of success, especially with regard to open-ended evolution.

Bedau and Packard have developed not only elegantly simple statistical measures for long-term evolutionary dynamics [5, 6, 4, 9], but also a test for unbounded evolution [7]. The test is so adaptable that it can be applied to any evolving system with an available record of its components' existence times, such as the biosphere's fossil record. Any artificial system of agents that actively process and transform their environment can be tested, and those that have been include Tierra-like systems [1, 28], Echo [17], Bugs [23], Lindgren's model of evolving strategies in the iterated prisoner's dilemma [20], Arthur's "Bar Problem" [2] and Ecolab [26]. A variable-length strings variant of Dittrich and Banzhaf's self-evolving binary string system [15], and perhaps also Banzhaf's earlier system [3], would be further interesting subjects for the test. Prior to the system detailed in this paper, only taxonomic families in the fossil record have passed the test[1].

[*] Parts of this paper have been published in [12], [10] and [11].

The following quote from their discussion section summarizes Bedau, Snyder and Packard's conclusion. Class 3 systems are those with unbounded evolutionary dynamics (see section 2).

> "Other natural evolving systems probably show class 3 dynamics. Class 3 dynamics might even be detectable in systems like the global economy or Internet traffic. We also suspect that no existing artificial evolving system has class 3 dynamics. In our opinion, creating such a system is among the very highest priorities of the field of artificial life. From one perspective, this is a negative result: Echo, and perhaps all other existing artificial evolutionary systems, apparently lack some important characteristics of the biosphere – whatever is responsible for its unbounded growth of adaptive activity. But at the same time this conclusion calls attention to the important constructive and creative challenge of devising an artificial model that succeeds where all others have failed." [7, p. 236]

Recently Stout and Spector ([27]) have provided valuable further validation of the test (modified as described in this article). They conducted a series of experiments carefully designed in an attempt to achieve class 3 dynamics in "intuitively unlife-like" systems. The test proved robust against these attempts. In their conclusion, Stout and Spector noted that "with simple fitness functions there appears to be a trade-off between new activity (innovation) and persistence"; that both Bedau et al.'s "shadow" mechanism (see section 2) and the method of component activity normalization outlined in this article (see section 9) are of particular importance; and finally:

> "In order to achieve sustained innovation and persistence simultaneously the fitness function must be such that new innovation is always necessary (and possible), which implies a function that is always changing, and yet also such that adaptive innovations remain adaptive. This is a tall order, and probably not achievable by any simple, "uninteresting" fitness function." [27, p. 142]

The remainder of this paper is structured as follows. Section 2 summarizes Bedau and Packard's statistical classification system for long-term evolutionary dynamics, and the resulting test for open-ended evolution. Sections 3 and 4 describe 'Geb', a system designed to verify and extend theories behind the generation of evolutionarily emergent systems, and its behavioral results. A complete description of the system is included here, to enable a full understanding of the implementation of the evolutionary statistics, as documented in section 5. Section 6 presents the result that, according to these statistics, Geb exhibits unbounded evolutionary dynamics, making it the first autonomous artificial system[2] to pass this test.

However, having passed it, the most prudent course of action is to look for weaknesses in the test. So the test is evaluated and criticized in section 7, most significantly with regard to its normalization method for artificial systems. Sections 8 and 9 present a modified normalization method, based on component activity normalization, that overcomes these criticisms. The results of the revised test, when applied to Geb, are reported and discussed in section 10. These indicate that this system does indeed exhibit open-ended evolution. Based on this discussion, section 11 concludes with a more confident, yet still cautious, evaluation of success and a direction for future work.

2. Bedau and Packard's evolutionary statistics

The motivation behind the statistics is to measure the level at which a system generates and then persistently uses adaptive innovations. By continually measuring both the generation of and the persistent use of adaptive innovations, these statistics are able to determine unbounded evolutionary dynamics, which requires both be ongoing.

Bedau and Packard's test is based on the following statistics (from [6, 7]), which are calculated from the record of components' existence times.

Activity increment (by presence).

$$\Delta_i(t) = \begin{cases} 1 & \text{if component } i \text{ exists at } t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

A component's activity is intended as a measure of its level of usage so far in evolution. An activity increment of one unit in each time step that the component exists satisfies this provided a component is only considered to exist if it is being used. So, for example, a gene that is present in a population but not currently expressed should not be considered to exist in the population of in-use gene components.

This is not the only activity increment that Bedau et al. have used, but it is the best for comparison across systems because it can be calculated for any system with a record of components' existence times.

Evolutionary Activity of a component.

$$a_i(t) = \begin{cases} \sum_{\tau=0}^{t} \Delta_i(\tau) & \text{if component } i \text{ exists at } t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Diversity (the number of components present, in use).

$$D(t) = \#\{i : a_i(t) > 0\} \tag{3}$$

Total cumulative evolutionary activity ('total activity').

$$A_{\text{cum}}(t) = \sum_i a_i(t) \tag{4}$$

Mean cumulative evolutionary activity ('mean activity').

$$\bar{A}_{\text{cum}}(t) = \frac{A_{\text{cum}}(t)}{D(t)} \tag{5}$$

Total activity provides a measure of the extent of evolutionary activity: how much activity has accumulated and remained in the system through the persistent use of components. It can be unbounded due to unbounded mean cumulative evolutionary activity, unbounded diversity, or both.

Table I. Classes of evolutionary dynamics and their statistical signatures, based on table 1 from [7][3]. Rows 3b and 3c have been added to class 3.

		Statistical Signature		
Class	Evolutionary Dynamics	D	A_{new}	$\bar{A}_{\text{cum}}$
1	none	bounded	zero	zero
2	bounded	bounded	positive	bounded
3a	unbounded (D)	unbounded	positive	bounded
3b	unbounded ($\bar{A}_{\text{cum}}$)	bounded	positive	unbounded
3c	unbounded (D & $\bar{A}_{\text{cum}}$)	unbounded	positive	unbounded

New evolutionary activity per component ('new activity') is defined by a lower activity bound a_0, above which a component can be considered adaptively significant, an upper activity bound a_1, below which a component can be considered 'new', and equation 6:

$$A_{\text{new}}(t) = \frac{1}{D(t)} \sum_{i:a_i(t)\in[a_0,a_1]} a_i(t) \tag{6}$$

New evolutionary activity (per component) provides a measure of the intensity of evolutionary activity: the level at which the system is generating adaptive innovations. Unbounded evolutionary activity requires both ongoing intensity and increasing extent of evolutionary activity: both positive new activity and unbounded total activity.

For A_{new} to be a good measure of new activity, the range $[a_0, a_1]$ should be chosen such that component activities within it can be considered both adaptively significant and not among the highest activities. So a_0 should be high enough to screen out most non-adaptive activity, and a_1 should be low enough that a good proportion of activities lie above it.

For artificial systems, a "shadow" should be run, mirroring the real run in every detail except that whenever selection (artificial or natural) operates in the real system, random selection should be employed in the shadow. The statistics from this shadow run can then be used to determine a_0 and levels of total and mean activity that can be considered adaptively significant.

After determining long-term trends in these statistics, the system being examined can be classified according to table I. The hallmark of class 3 (unbounded evolutionary dynamics) is unbounded total cumulative evolutionary activity ($D * \bar{A}_{\text{cum}}$) in combination with positive new evolutionary activity per component[4]. This provides one possible definition of and test for open-ended evolution.

Other possibilities exist with zero A_{new}, but these belong in class 1 (no evolutionary activity) because such cases have no significant new components. Table 1 in [7] only shows the first row (3a) for class 3, but footnote 1 in that paper acknowledges the other rows (3b and 3c). Note that table I includes all possibilities for positive A_{new}, because zero $\bar{A}_{\text{cum}}$ implies zero A_{new}. So any system with unbounded evolutionary dynamics will belong to class 3 (one of 3a, 3b and 3c).

3. The Geb model

Geb[5] is a system of agents that actively process and transform their environment. It was designed to verify and extend theories behind the generation of evolutionarily emergent systems. Its most important design features are a sufficiently correlated genotype to phenotype mapping, which makes it suited to long-term incremental evolution, and most importantly that it is a purely biotic selection system: *all* selection results from parts of the system (segments of code) carrying out (inter-)actions which are evolvable, either directly (the actions can evolve) or through their activation (their controller can evolve). In Geb, *all* selection results from agent interactions which are activated by a genetically specified controller. Details of the theory behind both Geb's overall design and the choice of each subsystem are beyond the scope of this paper but can be found in [13]. This section contains a complete description of the system, to enable a full understanding of how Bedau and Packard's evolutionary statistics have been implemented.

The agents operate within a simple two-dimensional toroidal virtual world (figure 1) which is divided into a 20×20 grid of squares. No two agents can occupy the same square at any one time – no square in figure 1 contains more than one individual's center. This effectively gives the agents 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 agent has a forward (facing) direction, set randomly at birth. Agents are displayed as filled arcs, the sharp points of which indicate their direction.

Each agent has a genetically specified neural controller. The inputs to this neural network are taken from the outputs of other (nearby) agents' neural networks, so allowing agents to process and transform their environment. Agents also transform their environment by moving around in the world, reproducing with and killing other agents. These actions are determined by outputs from the agents' controllers.

This is Geb's main algorithm:

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

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

These are the steps involved for each individual:

1. Network inputs are updated. See section 3.2.
2. Development – an iteration of the ontogenesis mechanism. See section 3.3.
3. All neural activations, including network outputs, are updated. See section 3.1.
4. Actions associated with certain network outputs are carried out according to those outputs. These actions are reproduce, fight, turn counterclockwise, turn clockwise, and move forward. See section 3.2.

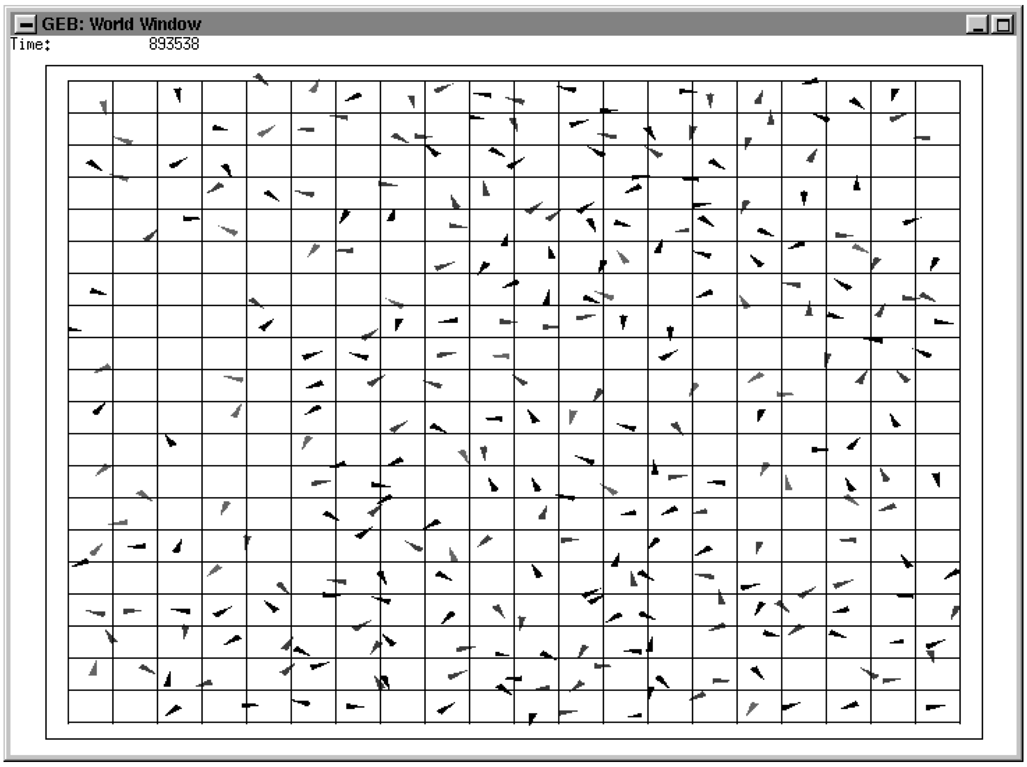


Figure 1. The experimental world (Geb).

3.1. Geb's neural networks

The artificial neural networks used in Geb are recurrent networks of nodes as used successfully by Cliff, Harvey and Husbands in their evolutionary robotics work [14, 16, 18]. The neural model (figure 2) is based on McCulloch and Pitts' original (1943) proposal [22], which includes a distinct inhibitory mechanism. Cliff et al. evolved recurrent networks of these nodes for visual navigation tasks in simple environments. Such networks were chosen (rather than alternative neural network designs such as feed-forward perceptron-based networks) for use in Geb because of their suitability for adaptive behavior work, as demonstrated by Cliff at al.'s experiments. All links have unit weight and no lifetime learning is used.

Each node has a bit-string 'character' (label) attached to it, used to match agents' 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.

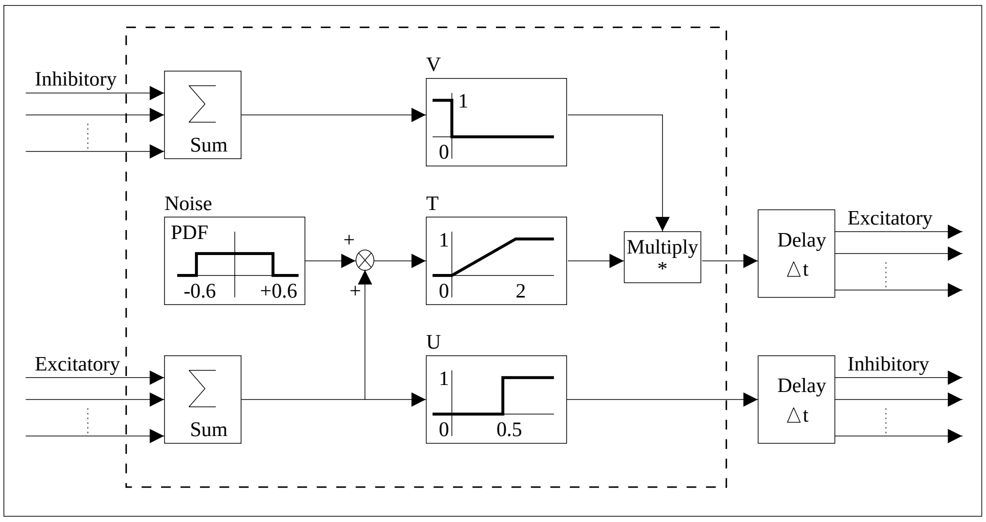


Figure 2. Schematic of a neuron, from [14].

3.2. Agent—environment interactions

There are five built-in actions available to each agent. Reproduce and fight (kill agent in front if there is one) were chosen to enable purely biotic selection: either is a sufficient action for biotic selection, but both are required if there is to be no abiotic selection. The remaining three actions allow basic locomotion, intended to enable complex behaviors that could be interpreted by Geb's user. Each action is associated with network output nodes whose characters start with a particular bit-string:

1. `01...` Try to *reproduce* with agent in front
2. `100...` *Fight*: Kill agent in front (if there is one)
3. `101...` *Turn counterclockwise*
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 agent 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 the relevant network outputs.

When an agent 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 agent cannot reproduce with an individual that is fighting if this would involve replacing the fighting individual.

Reproduction involves single point 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, which occurs at each reproduction, is a single bit-flip on the child genotype.

An agent'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 neighborhood. 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. For the runs reported in this paper, the input neighborhood is a 7×7 square area centered on the relevant agent.

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.

3.3. Developmental system

A class of L-systems with context-free production rules was designed for the evolution of networks of the neurons outlined above. Design of the developmental system was carried out with both genetic neutrality and developmental requirements in mind. Specific attention was paid to producing a system in which children's networks resemble aspects of their parents'. A genotype determines the L-system's production rules which determine the agent'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. Because computer memory is finite, it is necessary to limit the number of links. This is achieved by imposing limits on the number of nodes (20) and developmental steps (99, each step processing every node).

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 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 an 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`) is matched by the reproduction character. The network input node's character (left input `01`) matches this output without matching or being matched by the other nodes, and without matching any of the other action characters. Finally, the hidden node's character neither matches nor is matched by the other nodes' or the action characters:

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

Development takes place throughout the individual's life. Figure 3 provides an example of the development of a very simple network. It shows the axiom (age 0) network growing as, at every time step, each node has the most specific rule applied to it. Only the first two steps are shown.

3.4. Genetic decoding

The genetic decoding of production rules is loosely similar to that in [8]. 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. This method is well suited to a system in which genotype length changes through the addition or removal of sections (here just one bit position per reproduction – see section 3.2) from within the genotype.

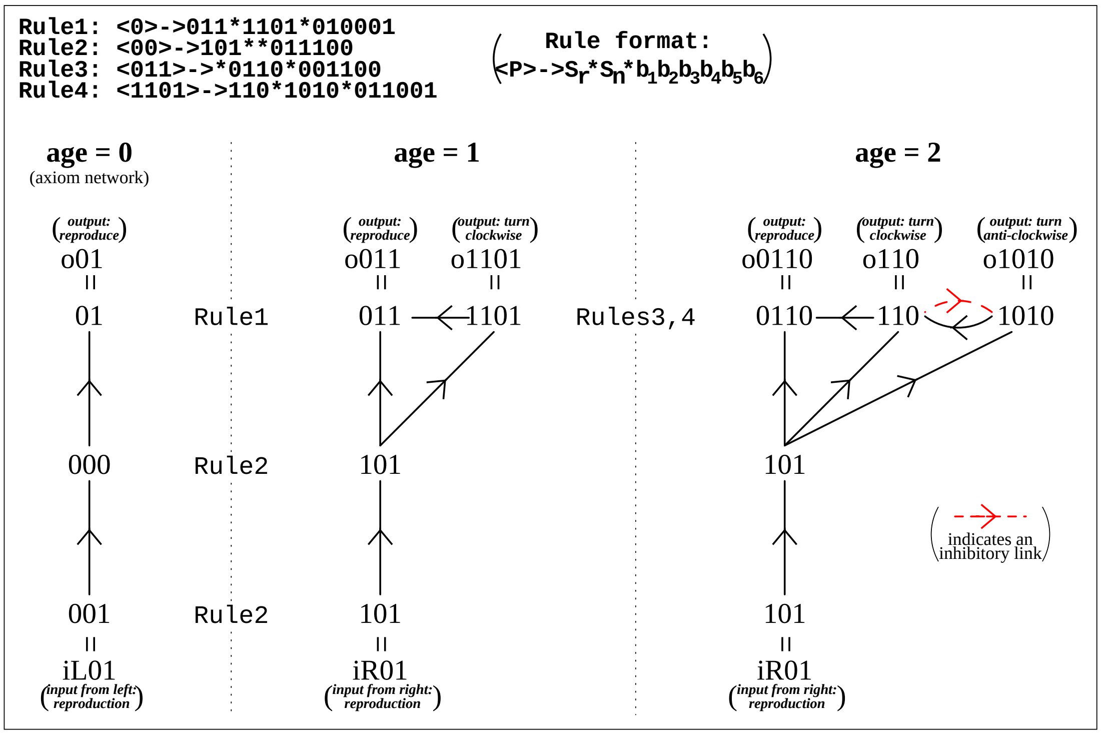

Rule1: <0>->011*1101*010001
Rule2: <00>->101**011100
Rule3: <011>->*0110*001100
Rule4: <1101>->110*1010*011001
Rule format: <P>->S_r*S_n*$b_1b_2b_3b_4b_5b_6$
age = 0
(axiom network)
age = 1
age = 2
output: reproduce
o01
01
000
001
iL01
input from left: reproduction
Rule1
Rule2
Rule2
output: reproduce
o011
output: turn clockwise
o1101
011
1101
101
101
iR01
input from right: reproduction
Rules3,4
output: reproduce
o0110
output: turn clockwise
o110
output: turn anti-clockwise
o1010
0110
110
1010
101
101
iR01
input from right: reproduction
indicates an inhibitory link

Figure 3. Example neural network development.

```
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 4. Example rule generation.

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 start over at the next bit position, 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 4 shows an example. As the first two bits are 11, an attempt is made to read a rule from the third bit onward. The segment-reading procedure above is carried out for the predecessor, which is empty because the third genotype bit is a 1; for the first successor, which has one bit 1 from the fifth genotype bit because the fourth genotype bit is a 0, but only one bit because the sixth genotype bit is a 1; and for the second successor, which has one bit 0 from the eighth genotype bit because the seventh genotype bit is a `0`, but only one bit because the ninth genotype bit is a `1`. Having read these three characters, the six link-details bits are read, in this case from the tenth genotype bit onward. As the last link-details bit can be read, that is reading has not gone beyond the end of the genotype, a rule has been successfully read and is added to the list of decoded rules. This rule-reading procedure is now carried out again, starting from the second genotype bit. As the second and third genotype bits are `11`, an attempt is made to read a rule from the fourth bit onward. This attempt is again successful, as the sixth link bit can be read without going beyond the end of the genotype. As the third and fourth genotype bits are not `11`, and likewise for the fourth and fifth genotype bits, no attempt to read a rule will be made for these positions. The fifth and sixth genotype bits are `11`, so an attempt is made to read a rule from the seventh bit onward. However this and subsequent attempts fail because the procedure runs out of genotype before completing a successful read.

After reading all possible rules from a newborn'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 such (currently) superfluous rules, which constitute the vast majority of decoded rules from long genotypes, are filtered out, much reducing memory required by Geb. This process does not affect the genotypes.

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 rule ii from the figure 4 example would not be added. 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.

4. Behavioral results

The system has consistently produced certain macro-level behaviors, including those below, although obviously the details of its evolution are different for every run. This makes it difficult to describe the behavior except by focusing on some typical and interesting observations.

4.1. 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 forward and kill* dominate the population. Some networks do better than others but not sufficiently well for them to display a sole dominating effect on Geb's world window.

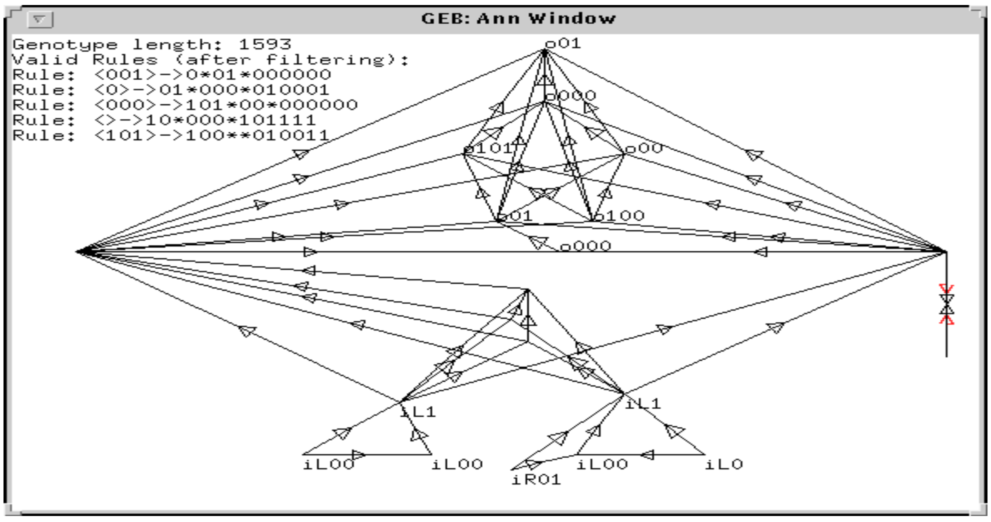

```
Genotype length: 1593
Valid Rules (after filtering):
Rule: <001>->0*01*000000
Rule: <0>->01*000*010001
Rule: <000>->101*00*000000
Rule: <>->10*000*101111
Rule: <101>->100**010011
```

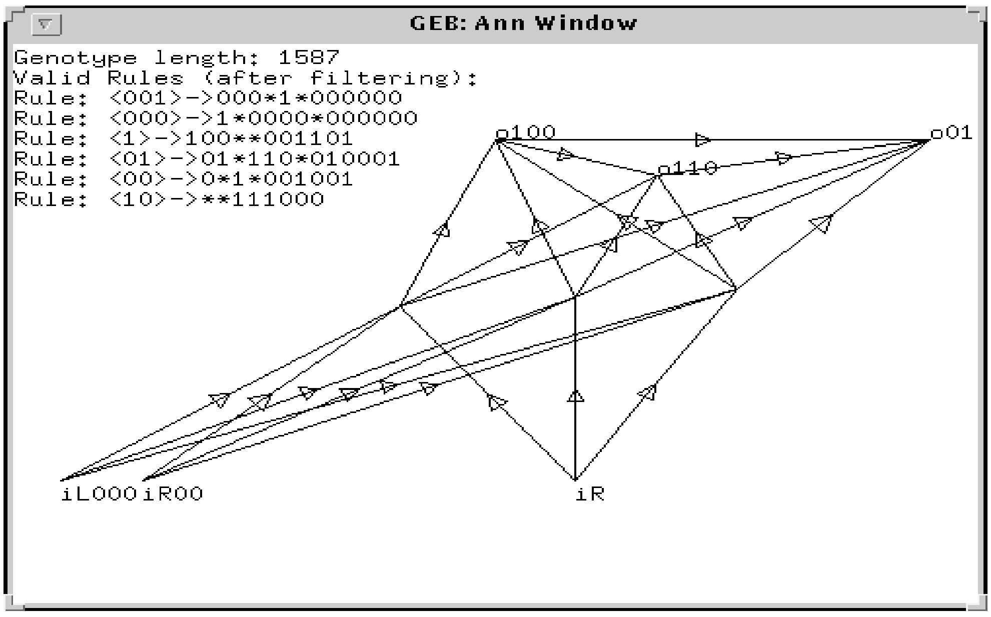
Figure 5. A dominant agent's neural network.

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 5 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 counterclockwise, 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 agent 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.

4.2. Naturally arising coevolution

Because of the success of this first dominant species, especially their success at killing other agents, the world always has enough space for other agents to exist. Such other agents tend not to last long: almost any movement will bring them into contact with one of the dominant agents, 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 5) is shown in figure 6. 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



```
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 6. A rebel agent's neural network.

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 toward 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 where it responds best. Most rebels have more complicated networks, which are very difficult to comprehend. Indeed, many of these have proved unassailably difficult to understand in detail.

4.3. Initial indications of ongoing coevolution

Figure 7 shows *running averages* of the number of agents reproducing and killing, from a typical experimental run. Each point is the average of the raw data (number of appropriate agents) over a window moving along the time axis. This filters out Lotka-Volterra population cycles and short term random variations, revealing long term shifts. This figure, and others like it from other runs, suggest that further species emerge, indicating ongoing evolutionary emergence. However, agents 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. A more statistical approach is required in order to test for ongoing evolution.

5. Implementation of Bedau and Packard's statistics in Geb

In order to apply Bedau and Packard's test to an evolutionary system, the biggest decision to make is what the class of components should be. As Geb's genotypes both change length and contain a high degree of neutrality[6] the genotype is not a

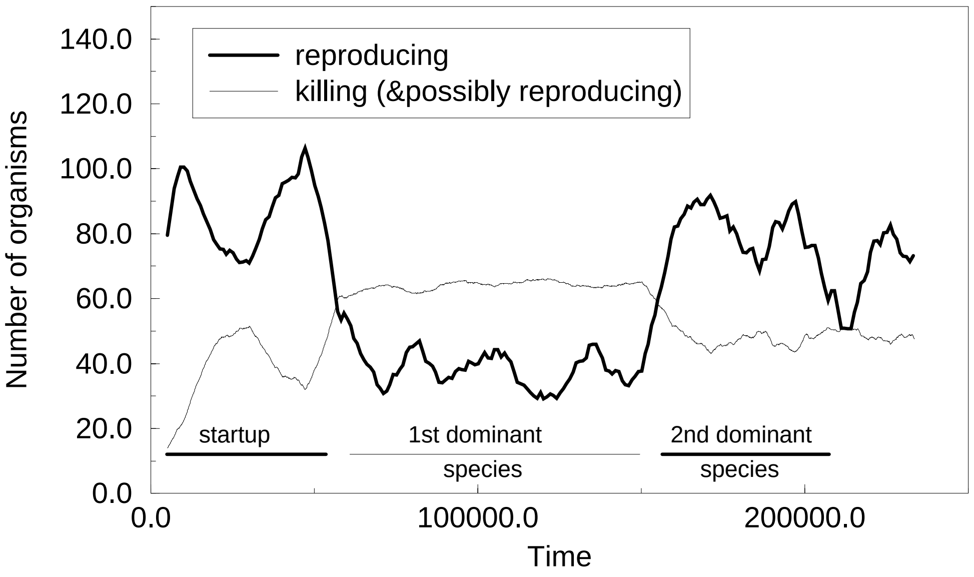


Figure 7. Running averages of population sizes by actions, from a typical run.

good choice of component class. Production rules, the alleles from a genotype, are a much better choice. It can be expected that if a production rule has an adaptive advantage, then it will persist. So the production rule would be a suitable choice of component. Better still is the choice "production rules that survive the filtering process at birth", for these are the rules that are actually used in the developmental process (section 3.4); the idea behind activity statistics is to measure the degree to which components both *persist* and are *used*.

When a mutation causes a component to be lost from the population (of production rules that survive the filtering process), the activity count of the original component is no longer included in the total activity of the system, even if the mutation is functionally neutral. At first I implemented the activity statistics on production rules directly. But there is often a high degree of neutrality in a production rule, especially when its 'successors' relate to neurons that are over-specified (have excess bits at the end of their characters) or development-terminal (not matched by any production rule). The predecessor and link-bits sections of production rules are less mutable. If a predecessor bit is mutated, then the rule will most likely either fail to match or be less specific to its target neuron than another rule. If a link detail bit is mutated, then the result will more often than not be a damaged network, and agents with that production rule active (not filtered out) will be driven to extinction. So the choice of component used here is 'predecessor plus link details' $(\mathcal{P}, b_1, b_2, b_3, b_4, b_5, b_6)$. This can be thought of as a disjoint grouping of alleles, with each group being a component. Which individual a component is from is irrelevant: two identical production rules in two different agents result in two instances of the same component.

This grouping does not completely remove the neutrality problem. As successor lengths increase, neural character lengths increase, and so the number of predecessors that can potentially match a typical neural character increases. If two rules have the same successors and link-details (or neutral variants), then it makes no difference to development which one is used. So, as component lengths increase, we can expect the level of neutrality to increase.

Having chosen this component class, there is a clear consequence for the possible classifications of evolutionary dynamics. Because the number of neurons that an agent can have is limited (for practical reasons), the number of production rules that can survive filtering is limited. And because the population size is small (a maximum of four hundred agents), there is little room for more than a couple of species at a time. So diversity of these components will certainly be bounded, and we can rule out class 3a and 3c dynamics.

5.1. Implementation details

Geb's shadow mirrors the real run in every detail except that selection is random. Whenever a real agent is killed, a randomly chosen shadow agent is also killed. Whenever a real agent is born (as the product of two real agents), a new shadow agent is born as the product of two randomly chosen shadow agents, using the same reproduction procedure with the same rates of crossover and mutation. For each initial real agent born with single-bit genotype '0', an initial shadow agent is also born with single-bit genotype '0'.

Gathering evolutionary activity statistics, or rather the component existence record from which they are calculated, is (processor-)time consuming. It is not feasible to gather the statistics at every time step. So snapshot existence records are taken at regular intervals and the evolutionary statistics are calculated from these. In the results reported here, snapshots were taken every one thousand time steps. To put this in context, the run reported lasted six million time steps, during which time there were over five hundred and eighty million agent reproductions. In each time step, every agent is updated. Because activity is intended as a measure of how much a component both is used (already covered above) and *persists*, I screen out (in each of the real and shadow populations) isolated occurrences: when a component occurs in the current snapshot but not the previous one.

In the previous section, total extinction (population size dropping to one individual) was not mentioned because it had not been encountered. However, there was no mechanism in place to prevent it and, during the long trial runs undertaken when experimenting with evolutionary statistics, I encountered occasional runs in which total extinction occurred. So for the set of runs from which the example reported here is taken, I set a minimum number of agents to twenty. The fact that total extinction is so rare despite the population size being so small (a maximum of four hundred agents at any one time) indicates that there is no serious problem here. Once population sizes can feasibly be increased, the problem should in practice disappear rapidly.

6. Results and Discussion I

This section contains the results from a typical run, drawn from the full set of twenty runs referred to in the previous section[7]. Atypical variations are discussed at the end of this section.

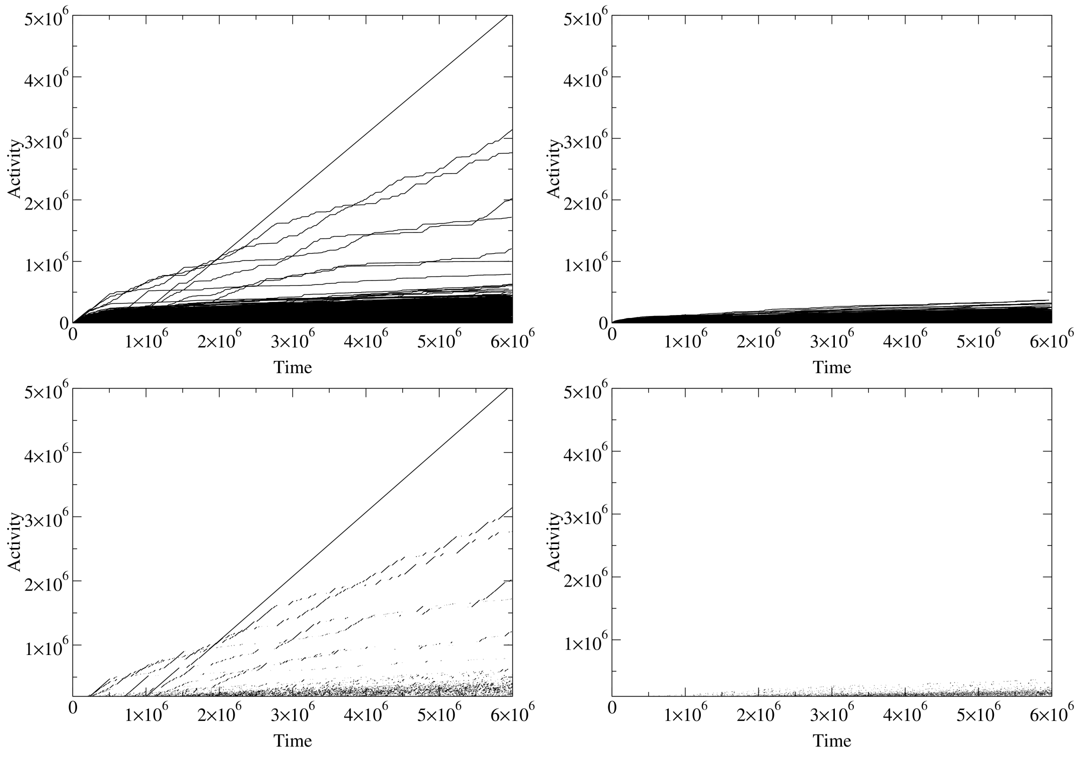


Figure 8. Activity wave diagrams for the real (left) and shadow (right) runs. The diagrams on the bottom have had all horizontal (no-increase) lines removed.

6.1. Activity waves

In order to gain an understanding of the dynamics behind the higher level evolutionary statistics, it is a good idea to look first at the activity wave diagrams, which simply show all components' activities plotted against time. Figure 8 shows the activity wave diagrams for the real and shadow populations.

The most obvious feature of the real run's activity waves in figure 8 is that many of them keep increasing. This would also be true in a similar analysis of genes from the biosphere's evolution. Genes that are beneficial to life tend to become basic for many species: humans have a significant proportion of genes in common with mice, flies and even plants. So because the components here are (groups of) genes, not whole genotypes, this feature does not imply a quasi-stable ecosystem.

In systems without neutrality new components initiate an activity (by presence) wave that increases with a constant slope and then stops when the component goes extinct. Here however, that increase is often shared between two or (perhaps many) more phenotypically equivalent components, with interchanging presence. If the population size was larger, then there would be greater scope for more than one of these neutral variations to be expressed in the population at any time. But with a population of less than four hundred, and short lifetimes, genetic variation spreads quickly through the population so wave transitions between neutral variants show up almost as either-or events.

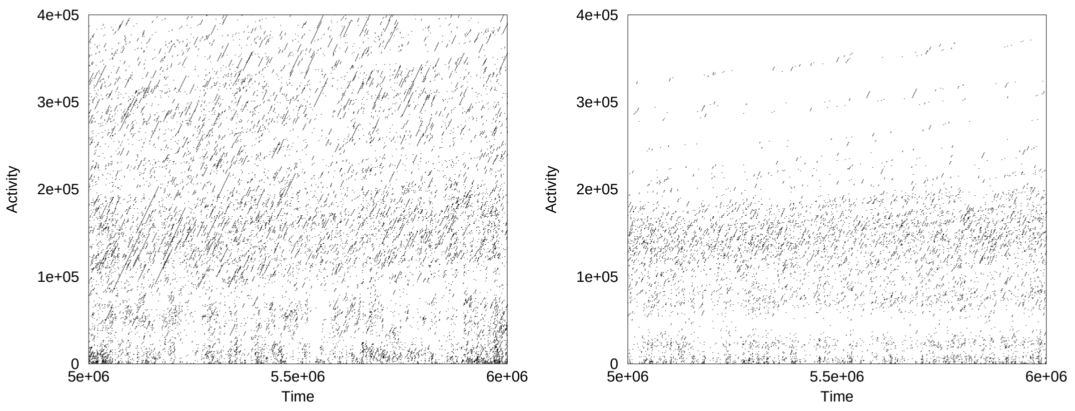


Figure 9. Activity point-plots for the real (left) and shadow (right) runs in the last million time steps, within the shadow's activity range.

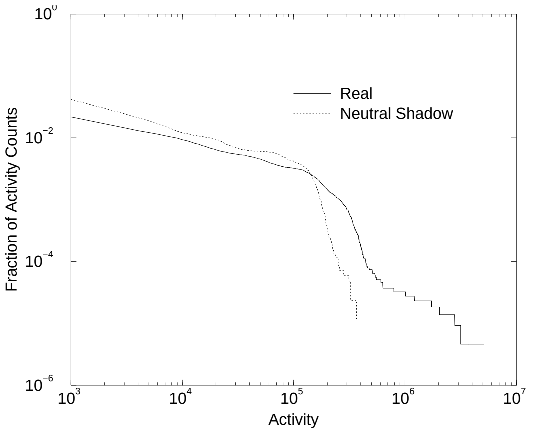


Figure 10. Log-log plot of the component activity distributions.

Most activity falls within the solid black regions at the bottom of each graph. So it is instructive to look in more detail at the bottom-right corner of the activity wave diagrams. Figure 9 shows the activity waves in the last one million time steps, with just a point for each recorded activity value. Its scale covers the shadow run's full range only, so that the real and shadow data can be easily compared. Notice the long runs of consecutively increasing activity in the real run, and the lack of them in the shadow.

6.2. Determining the new-activity range

In order to measure new activity (A_{new}), we must first determine the range $[a_0, a_1]$ of component activity values that should be considered both adaptively significant (so a_0 should be high enough to screen out most non-adaptive activity) and not among the highest activities (so a_1 should be low enough that a good proportion of activities lie above it). The method given in [7] involves finding the activity value which is equally likely to have occurred in the real run as in the shadow run, and setting $[a_0, a_1]$ to be a narrow band that surrounds it. Figure 10 shows the component activity distributions for the run reported here. These cross at an activity of approximately

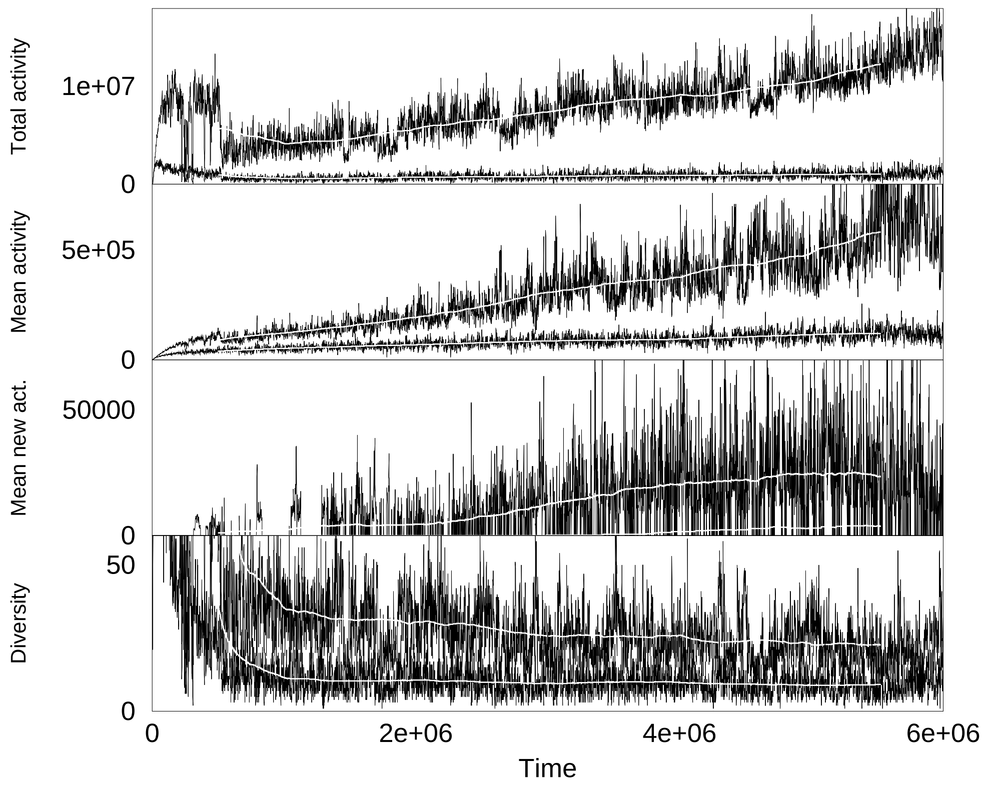


Figure 11. Total activity, mean activity, new activity and diversity from a typical Geb run and its shadow. Running averages are shown in white. In all four graphs these are higher in the Geb run than in its shadow.

1.42×10^5. However, it is clear from figure 8 that the component activity distributions are far from constant over the run, and that this value increases during the course of the run. Looking again at figure 9 shows that by the end of the run, activities of approximately 1.42×10^5 are common in the shadow.

The best approach is probably to recalculate the range $[a_0, a_1]$ in each time step (and perhaps apply a running average to smooth it), because we are working with a system with increasing component activity values; this would also produce the most impressive results for A_{new}. However, in order to avoid the criticism that I have changed the method, I have chosen here to use a fixed range that screens out the majority of the shadow activity in the final million time steps of the run. This results in artificially low values for A_{new} early on in the run, but the results are still positive despite this. Looking at figure 9 we can see that most of the shadow's activities are below 3×10^5. In fact even in the last million time steps, less than 3.5% of the shadow activity is above 2.8×10^5. Further, it is also clear from figure 9 that an activity of 3×10^5 is far from uncommon in the real run. In fact approximately 27% of the real activities are above 3.2×10^5 in the last million time steps. So the results that follow were calculated using a new-activity range of $[2.8 \times 10^5, 3.2 \times 10^5]$.

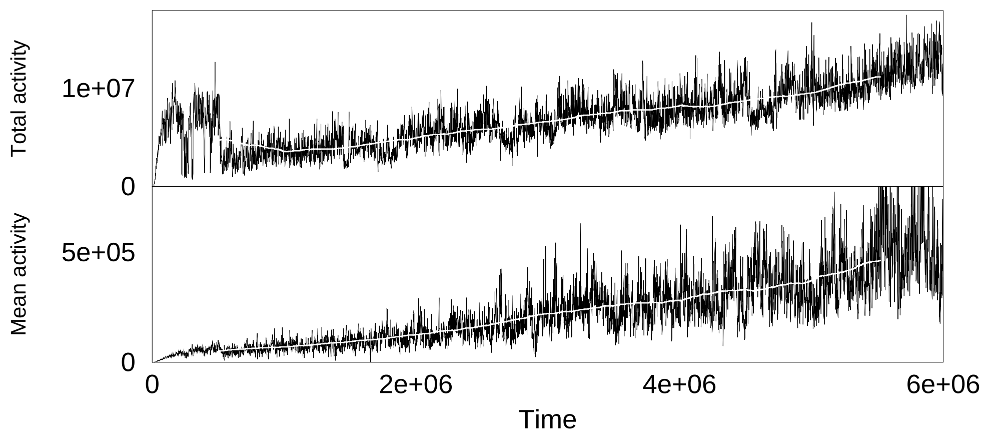


Figure 12. Normalized total and mean activity. Running averages are shown in white.

6.3. Evolutionary statistics and classification

Figure 11 shows both total and mean activity increasing rapidly in the real run, and much slower in the shadow run. New activity is positive in the real run, and much higher than in the shadow, which exhibits only occasional blips of new activity. Diversity is bounded in both the real and shadow, as expected (see section 5).

Figure 12 shows the activity difference between the real and shadow statistics ("normalized activity"), for both total and mean activity. According to this classification system, these results clearly fall into class 3b: unbounded evolutionary activity.

6.4. Atypical runs

These results are typical of the twenty runs that were carried out for this set of experiments. However, three of the runs encountered problems, causing their results to be atypical. Two of these effectively met total extinction. In section 5.1 I mentioned that I imposed a minimum limit on the number of agents, in an attempt to avoid total extinction. However, if population size hits this limit and does not increase rapidly, then many reproductions may occur with selection effectively random. This causes evolutionary activity to plummet as adaptive traits are lost. Once lost, this activity cannot be regained, except by the evolution of new adaptive components. These results should not be a cause for concern, for the same reasons mentioned in section 5.1: once population sizes can feasibly be increased, the problem should in practice disappear rapidly.

In the third atypical run, it appears that a freak mutation has caused the only existing species to take on a behavior of never reproducing or moving forward and always turning and trying to kill. Of course this would ordinarily be a very poor strategy. It is easy to imagine how the bad gene (production rule) could have spread through a population of just one species as fit individuals reproduced with the new unfit ones, causing their children to pick up the dominant bad gene. However, one would not expect this to pose a threat to a different species. This is easily verified: introducing just a few agents from any of the other evolved populations (from the other runs), causes the old agents to be rapidly displaced by the newcomers. So this

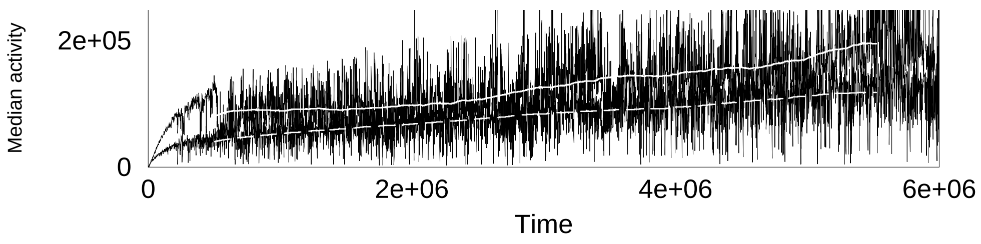


Figure 13. Median activity from a typical Geb run and its shadow. Running averages are shown in white: solid for the real run, dashed for its shadow.

result is also not a cause for concern, for the same reason: it is due to the small population size, which cannot support more than one or two species at a time.

7. Evaluation of these results and the test

Geb has demonstrated class 3 behavior, and so passed the test as it stands. Does this mean that Geb truly exhibits unbounded evolution? Possibly, for it was designed to verify and extend theories of evolutionarily emergent systems generation and so a number of potential pitfalls have been avoided. However, having passed the test the most prudent course of action is to look for weaknesses in the test.

My main concern at this point is that the test relies on normalization (or validation) from a shadow that can drift away from core aspects of the real run that it is intended to shadow. For example, the components that exist in the real population at any one time (well into evolution) are almost certainly more densely clustered than those in the shadow. This example is an obvious concern in Geb, but the point is that any drift of the shadow away from the real run is a danger to the validity of the normalization process: The intention is to integrate the level of activity increments that occur above (or rather minus) the level that would have occurred under random selection, so we need to know the activity increments that would occur under random selection on our real system as it changes. Once the real and shadow populations have been allowed to evolve, we are no longer comparing the real run with a true shadow. The longer the period since the shadow was initialized to match the real run, the less relevant the shadow is to the real run. One way around this problem would be to develop a method of comparing the real run with a shadow that is regularly reset (both components and activity history) to be identical to the real run but which evolves using random selection between resets. Normalized activity increment between resets could then be determined by comparing real and shadow increments.

My other criticism of the test as it stands is in its use of mean activity when looking for unbounded activity growth, especially when classifying a system as belonging to class 3b. When diversity is bounded, the retention (forever) of a single component results in unbounded mean activity. The test should not be so influenced by such components, and should rather look for trends in typical components. So it is median activity, not mean activity, that should be measured, and required to be unbounded for a system to be classified as within class 3b. The activity waves from Geb's runs indicate that median activity is also unbounded. However, when median activity is measured in both real runs and their shadows, it shows up as unbounded in both

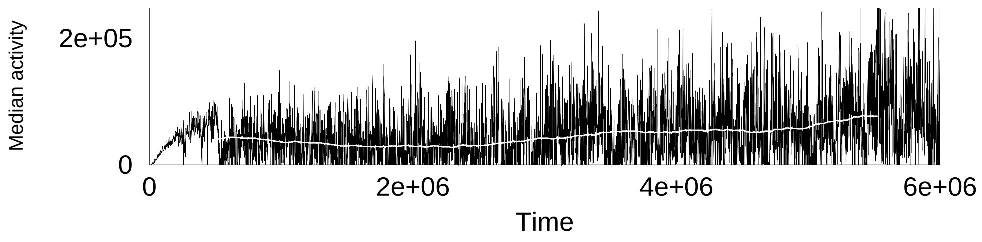


Figure 14. Normalized median activity. A running average is shown in white.

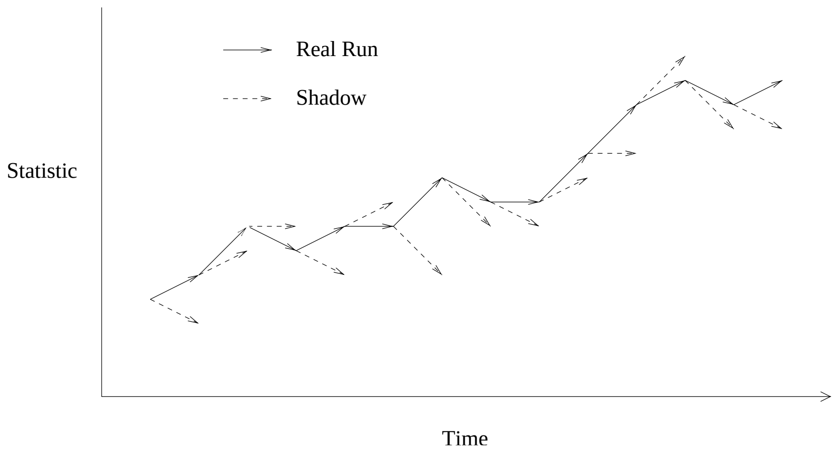


Figure 15. Illustration of the shadow-resetting method.

(figure 13), and it is not possible to make any firm conclusion (bounded/unbounded) about normalized median activity (figure 14). However, in light of the main concern above, just as the positive results cannot be trusted, this should not be seen as a cause for concern or fuel for further investigation along this path. The correct course of action is to proceed as outlined above, by developing a shadowing method that regularly resets the shadow state to the real state, and then look at the results again, including median activity.

8. The shadow-resetting method

Because there is reason to doubt a method of normalizing (or validating) evolutionary statistics that relies on a shadow that can drift away from core aspects of its real run, a new method is needed: one that regularly resets the shadow (both components and activity history) to be identical to the real run.

The basic idea is that immediately after each snapshot (when an entry is made in the component existence record), the shadow run has its components reset to those of the real run. This allows us to compare inter-snapshot changes in activity in the real run with the changes we would expect from random selection, the result being an improved generic shadowing mechanism. When calculating evolutionary statistics (and indeed when recording component numbers), the shadow's history is taken to be that of the real run - see figure 15.

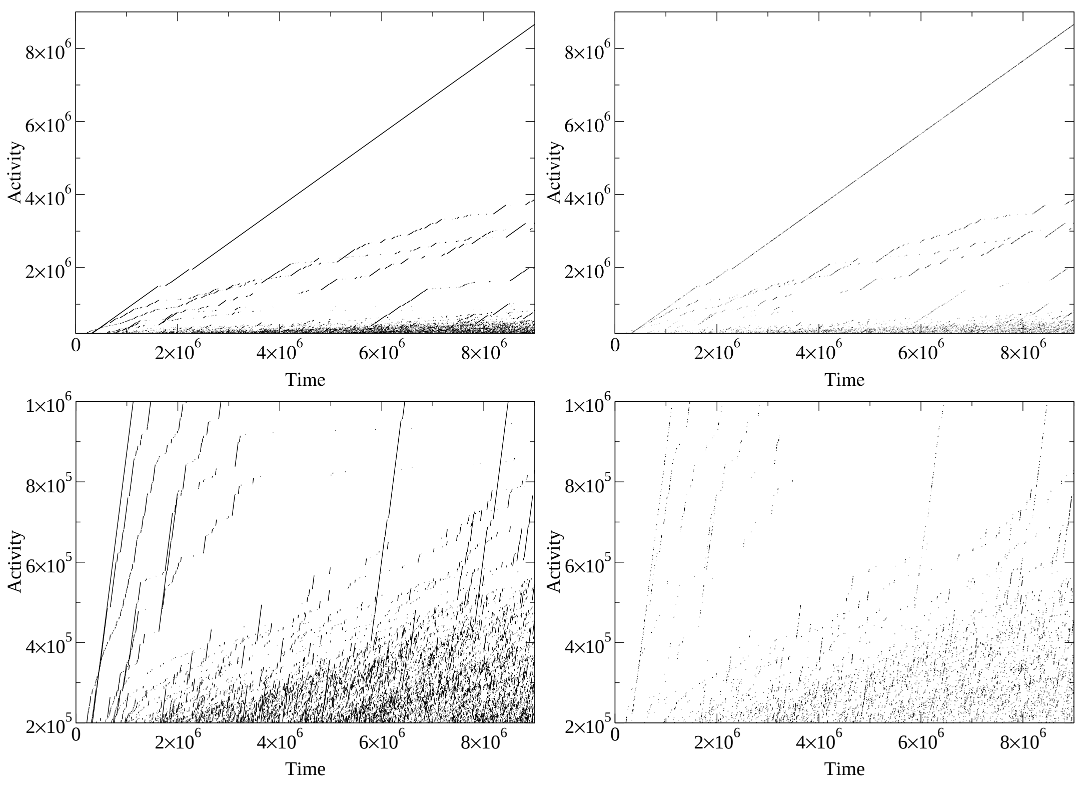


Figure 16. Activity wave diagrams for the real (left) and shadow (right) runs, with all horizontal (no-increase) lines removed. The lower diagrams show a magnified view of the activity range below 1 million.

Most of the results below are from a typical run, drawn from a set of twenty carried out using this procedure. Atypical variations found within this set are also reported and discussed. The component class used (production rule predecessor plus link details) is the same as in the previous sections, as are the run implementation details (snapshots every thousand time steps, isolated component occurrences screened out, minimum number of agents set to twenty). Figure 16 shows the raw real and shadow activity wave diagrams from the typical run. Shadow waves follow the real waves, because the shadow is reset after each snapshot. The shadow loses components between snapshots far more frequently than the real run does. This is especially true of the lower-activity components, as we should expect. Adaptively significant production rules have many redundant copies on a typical genome such that should mutation break the rule at one point, it will still be decoded from elsewhere on the genome. Such components can survive even sustained periods of random selection. Yet even the highest activity components are frequently lost in the shadow and this provides verification that the snapshot interval (one thousand time steps) is sufficient for comparing activity by presence. Employing a much higher reset frequency would require the use of an activity increment function which is sensitive to the number of occurrences of a component at any one time[8], for example $\Delta_i(t)$ = fraction of agents that have the i^{th} component at time t. However, because significantly more frequent snapshots are not feasible with the computational resources currently available, comparing activity by presence is still the best option.

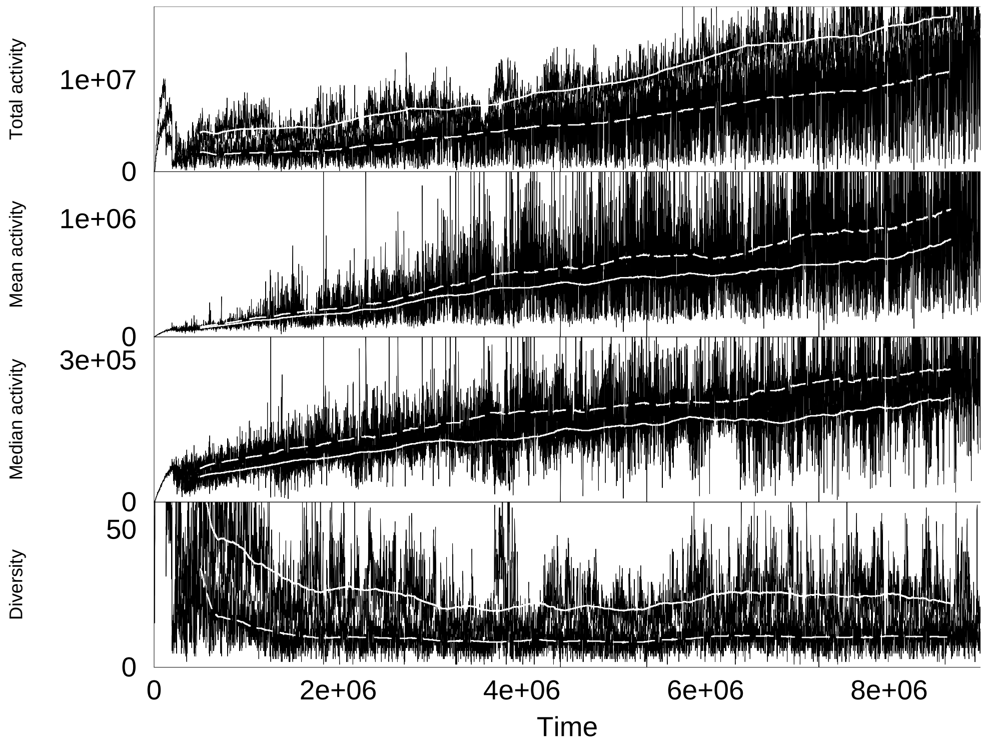


Figure 17. Total activity, mean activity, median activity and diversity from a typical Geb run and its regularly-reset shadow. Running averages are shown in white: solid for the real run, dashed for its shadow.

Of course it would not make sense to calculate activity statistics based solely on the shadow's component existence record. Perhaps the most obvious course of action (and the one I took at first) is to calculate activity statistics for the shadow on the basis that at each time step t we use the real run's component record for time steps before t, and the shadow's component record at t. Figure 17 shows the resulting statistics from the shadow, alongside the real run's statistics. Activity (total, mean and median) is unbounded in the shadow only because the shadow has its components reset to those of the real run after each snapshot. On average both total activity and diversity drop sharply in the shadow over each short (1000 time steps) interval after it has been reset to the real run's state. The shadow's mean and median activity statistics show that (on average) it is the higher activity components that remain in the shadow, in agreement with the discussion (above) of the activity wave diagrams. Do not be confused by the fact that mean and median activity increase in the shadow over each inter-snapshot interval. This is due to the loss of lower activity components, not the result of any increase in component activity.

These results are encouraging, but they provide no route to normalizing the real run's statistics in order to demonstrate a presence or lack of unbounded growth in, say, median activity. The method so far also provides no sound way of measuring new activity. So naively calculating the shadow's activity statistics leads us to a dead-end. The idea of resetting the shadow run's state to match the real run's state just after each snapshot is a good one, but how can it be used to normalize the real run's statistics?

9. Component activity normalization

The solution is to normalize at the lower level of individual components' activities, rather than at the component-population level. This is done by subtracting the shadow's component activity increment from the real run's component activity increment, for each component. So when calculating activity by presence, a component's normalized activity is incremented if and only if it persists (and is used) in the real run but not in the shadow, and is decremented if and only if it persists (and is used) in the shadow but not in the real run (although activity will still be read as zero for that time step). Here are the revised statistics:
Real run's component activity increment (by presence).

$$\Delta_i^{\mathrm{R}}(t) = \begin{cases} 1 & \text{if component } i \text{ exists in the real run at } t \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Shadow's component activity increment (by presence).

$$\Delta_i^{\mathrm{S}}(t) = \begin{cases} 1 & \text{if component } i \text{ exists in the shadow at } t \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Normalized component activity increment.

$$\Delta_i^{\mathrm{N}}(t) = \Delta_i^{\mathrm{R}}(t) - \Delta_i^{\mathrm{S}}(t) \tag{9}$$

Normalized component activity.

$$a_i^{\mathrm{N}}(t) = \begin{cases} \sum_{\tau=0}^{t} \Delta_i^{\mathrm{N}}(\tau) & \text{if component } i \text{ exists in the real run at } t \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Normalized diversity is defined by a lower normalized activity bound a_0^{N}, above which a component can be considered adaptively significant (see below), and equation 11:

$$D^{\mathrm{N}}(t) = \#\{i : a_i^{\mathrm{N}}(t) > a_0^{\mathrm{N}}\} \tag{11}$$

Note that this formula for D^{N} is only a suggestion for how diversity could be normalized when investigating systems with unbounded diversity: by counting the number of components whose normalized activity has passed the threshold at which we consider them to be adaptively significant (see below). This method of normalizing diversity is debatable. However, because Geb does not exhibit unbounded diversity I safely ignore that debate here, and do not calculate D^{N} for Geb. This is valid because no claim of unbounded diversity is being made, and because D^{R} (not D^{N}) is the relevant value to use when calculating $\bar{A}_{\mathrm{cum}}^{\mathrm{N}}$, $\widetilde{A}_{\mathrm{cum}}^{\mathrm{N}}$ and $A_{\mathrm{new}}^{\mathrm{N}}$, because D^{R} is the number of

components that contribute to those statistics.
Normalized total cumulative evolutionary activity.

$$A^{\mathrm{N}}_{\mathrm{cum}}(t) = \sum_{\substack{i:\,\text{component i exists}\\ \text{in the real run at t}}} a^{\mathrm{N}}_i(t) \tag{12}$$

Normalized mean cumulative evolutionary activity.

$$\bar{A}^{\mathrm{N}}_{\mathrm{cum}}(t) = \frac{A^{\mathrm{N}}_{\mathrm{cum}}(t)}{D^{\mathrm{R}}(t)} \tag{13}$$

Normalized median cumulative evolutionary activity.

$$\widetilde{A}^{\mathrm{N}}_{\mathrm{cum}}(t) = \operatorname*{Median}_{\substack{i:\,\text{component i exists}\\ \text{in the real run at t}}} a^{\mathrm{N}}_i(t) \tag{14}$$

Normalized new activity per component.

$$A^{\mathrm{N}}_{\mathrm{new}}(t) = \frac{1}{D^{\mathrm{R}}(t)} \sum_{i:\text{component i `new'}} a^{\mathrm{N}}_i(t) \tag{15}$$

See below for the details of calculating normalized new activity per component.

This is clearly the better approach, for it produces normalized component activities that measure how much each component's activity has increased above the increase that would have occurred had selection been random. So a component's normalized activity is a direct measure of the degree to which adaptive selection in the real run is causing the component to persist (and be used).

9.1. Determining the normalized new-activity criteria

The final requirement, before these statistics can be used to classify evolutionary dynamics, is a method of determining when a component is newly adaptively significant: Clearly the method from [7], as used in section 6.2, cannot be used with the revised shadowing mechanism. The method must provide a (normalized) activity level a^{N}_0 at which a component can be considered adaptively significant, and a procedure for dropping a component from the set of new components. For the second of these concerns, a simple upper bound cannot be used, because normalized activity can both increase and decrease, so a component could potentially be considered 'new' forever. The simplest (and adequate) solution is to consider a component to be 'new' (newly adaptively significant) in the snapshot at which its activity reaches a^{N}_0, and never after that. So each component is considered new at most once. This leaves the issue of determining a^{N}_0.

If the presence or absence of a component confers no adaptive advantage or disadvantage, then the real and shadow systems are equivalent for this component. Further, which is used as the reset-to system (after each snapshot) makes no difference to the component's activity. So the (normalized) activity distribution for this class of components will be symmetric about the origin. Therefore, provided we can make the assumption that the most negative normalized activity encountered during a run is

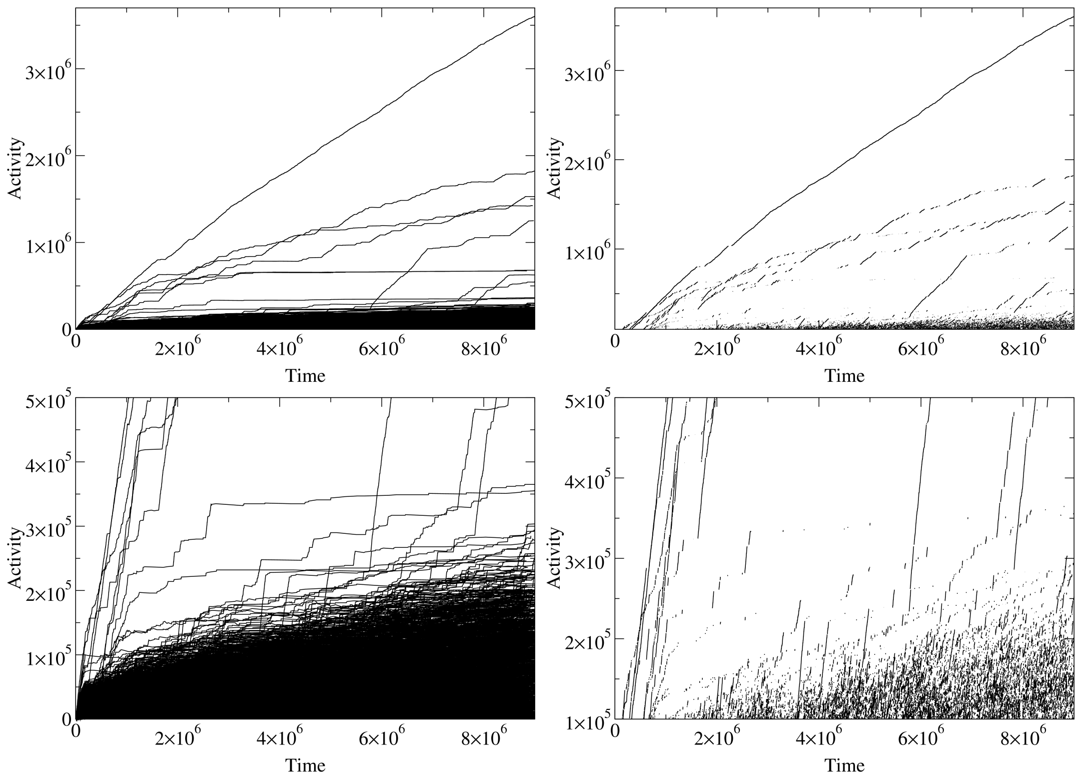


Figure 18. Normalized activity wave diagrams. Those on the right have had all horizontal (no-increase) lines removed. The lower diagrams show a magnified view of the activity range below 0.5 million.

from such a component, we can negate this value to find a level at which normalized activity can be considered adaptively significant. Even if this assumption does not hold, negating the most negative activity encountered provides a value above which activity can be considered adaptively significant, even if this bound is higher than necessary.

This method can be expected to work well when activity is calculated by presence (as it is in Bedau and Packard's test and so also here), where changes in component activity (Δ_i) are small when compared with the activities of non-adaptive components. However, we should not expect it to provide a good bound when calculating activity by, for example, usage (Δ_i = #components i at t), where the most negative activities arise from neutral mutations of high usage components, some of which a shadow will encounter before its real run does.

10. Results and Discussion II

This section contains the results from the typical run, already discussed above, drawn from the full set of twenty runs referred to in section 8. Atypical variations are discussed in this section.

Figure 18 shows the resulting normalized activity waves. Notice that the activity values are significantly lower than before normalization - see figure 16.

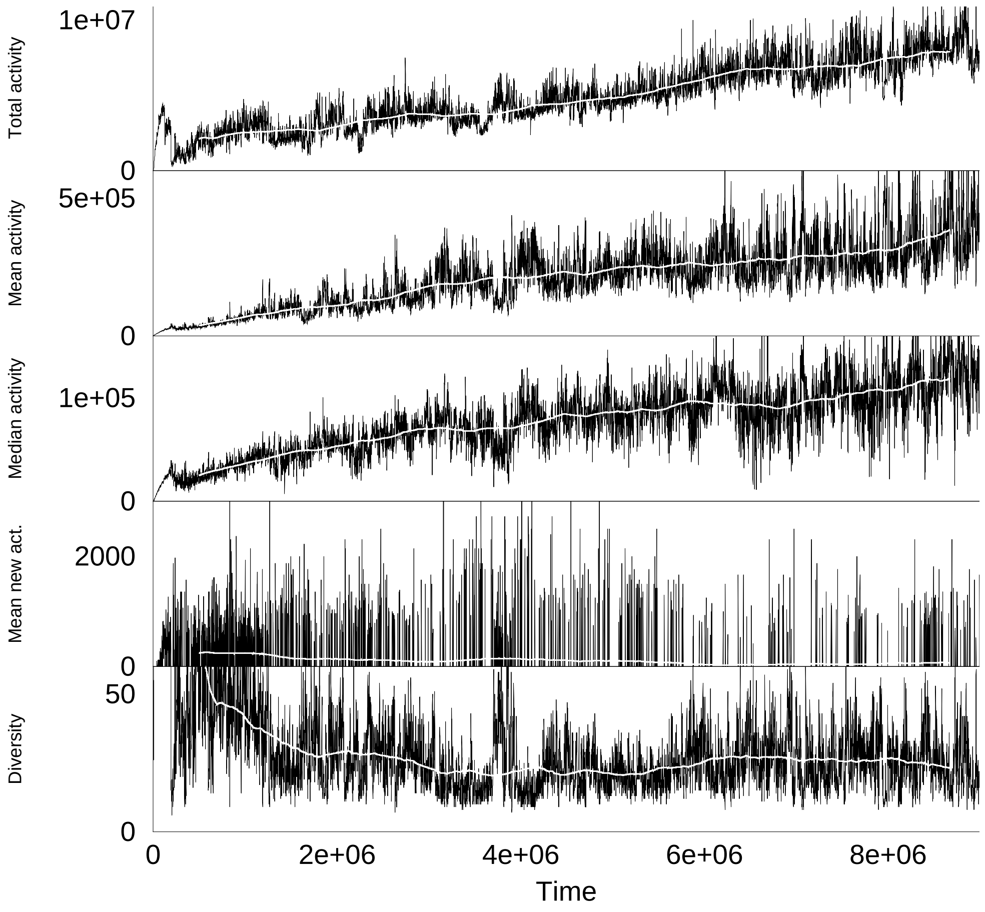


Figure 19. Normalized total activity, normalized mean activity, normalized median activity, normalized new activity, and real diversity. Running averages are shown in white.

In each of the twenty runs, the lowest normalized activity encountered was greater than -30, with -10 being a more typical value. For simplicity when calculating the evolutionary statistics for these runs, I used a new-activity threshold (a_0^{N}) of 30 on all runs. Figure 19 shows the resulting statistics for the typical run. Normalized median activity is unbounded (as are normalized total activity and normalized mean activity) and normalized new activity is positive. These results clearly fall into class 3b (according to this classification system): unbounded evolutionary activity.

These results are typical of the twenty runs that were carried out for this set of experiments. However, three of the runs effectively met total extinction, and two of the runs stagnated when the only existing species stopped reproducing. Both of these possibilities were encountered in the original test's results. As discussed there (section 6.4), they should not be a cause for concern.

It is easy to demonstrate that unbounded growth (or more accurately unbounded non-monotonic but *directed* increase) in activity, with positive new activity, is not a trivial consequence of unbounded genotype length. Consider the analogous system Geb^R, in which selection is random but all other details are as in Geb. Whenever a (randomly chosen) real agent is killed in Geb^R, a randomly chosen agent is also killed in its shadow. Whenever a real agent is born in Geb^R (as the product of two randomly chosen real agents), a new shadow agent is born as the product of two randomly chosen shadow agents. Of course running either real or shadow system from the same snapshot more than once would produce different results on each run, because of the stochastic nature of the systems. So normalized activity would unfold as a random walk, with 'step' probability distribution changing at each snapshot but always symmetric about zero.

Could unbounded growth in activity be a trivial consequence of unbounded genotype length in a biotic selection system that exhibits positive new activity? No, because the requirement remains that activity be retained, so that it can accumulate. For example, a (diversity-bounded) biotic selection system that continually generates new components only by mutation along (phenotypically) neutral networks would only be able to *use* a finite number of neutral variants at any one time. It would lose activity whenever a component is lost from (ceases being used in) the system.

11. Conclusions

Both criticisms of the original test (section 7) have been addressed. The revised shadowing method used here ensures that the normalization of statistics is through a shadow that remains true to its real run, and median rather than (or rather as well as) mean activity has been used in the classification.

Geb has demonstrated class 3 behavior: unbounded evolutionary dynamics. And this time we can have a greater degree of confidence in the results. However, this is a new variant of a previous test, and it is not beyond possibility that it could be improved upon. Certainty in these results can only come about through the application of the test to a range of evolutionary systems. That may take some time, since there are no other autonomous artificial systems that have passed even the original test. So for now we must be content with the conclusion that there is reason to believe that this system exhibits unbounded evolution.

While the caution of the previous paragraph is warranted, it is at least possible to say with certainty that these results qualitatively surpass those from previous artificial evolutionary systems. No previous autonomous artificial system has demonstrated unbounded evolutionary dynamics. As such these results provide validation of the theory behind Geb's design [13].

One significant weakness can be identified in Geb: it becomes difficult (impossible in practice) to understand or even identify novel behaviors as they emerge. With relatively simple agents, we can analyze their controllers (programs, neural networks or other) directly to determine the resulting behaviors. But as the complexity of evolved controllers increases, this becomes infeasible, especially when using neural controllers. We can only observe the resulting behaviors and attempt to identify innovation. The logical aim is therefore to develop future systems such that behavioral descriptions are as easy to generate as possible, probably by constructing the systems such that behaviors will be transparent to human observers.

Brain-body coevolution could be used to aid in the observation (identification and analysis) of emergent behaviors, enabling us to interpret behaviors from the motions of body parts. It would also have the advantage that interactions themselves would be directly evolvable, not just through their activation (controllers). Sims' evolution (by *abiotic* selection) of virtual creatures in a three dimensional physically simulated environment [24, 25] provides an excellent example of this. These experiments produced behaviors which are easy to identify and describe, operate over short distances but many time-steps in a non-simplistic way, and are largely the result of the evolved embodiment of the agents. Mainly as a result of increases in readily-available computational power, but also thanks in part to the availability of satisfactory physics engine packages, there have been a number of attempts to replicate Sims' work [29], although none of these have succeeded in producing such impressive results. Incorporating such features into a system such as Geb would certainly be a significant step forward, toward autonomous artificial systems with both unbounded evolutionary activity and behaviors that we can understand.

Notes

[1] Maley [21] makes the claim that two of his models, 'Urmodel 3' and 'Urmodel 4', exhibit unbounded evolutionary dynamics. However, Urmodel 3 shows less new activity than its shadow (with no reason to think that it would become greater), Urmodel 4 shows a lower mean activity than its shadow and both are only examined during their initial growth stages, so these claims are not valid. To my knowledge, there have not yet been any other claims of unbounded evolutionary dynamics in an autonomous artificial system.

[2] I use the term "autonomous artificial system" to refer to an artificial system with no ongoing human intervention, so discounting systems such as the global economy, Internet traffic and evolutionary systems such as Pfeiffer [19] in which fitness is derived from user evaluation or interaction.

[3] Bedau has since altered his class numbering scheme.

[4] Because positive new activity is required for classes 2 and 3, only systems that continue to generate new adaptations can be in these classes.

[5] The Geb system is available without charge for research use. It can be download from `http://www.channon.net/alastair/#Software`.

[6] In order to avoid confusion, I only use the term *neutral* to refer to genetic variations that are phenotypically equivalent, and not in relation to shadow runs.

[7] The two sets of runs reported in this paper, each consisting of twenty runs with shadows, consumed a total of approximately three years of 1GHz single-cpu time.

[8] Thanks to Mark Bedau for bringing this to my attention.

Acknowledgements

Thanks to Mark Bedau for discussion and many helpful comments on this work. Thanks also to the anonymous reviewers for constructive criticism and helpful comments, and to Wolfgang Banzhaf for his help with the paper.

References

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

2. W. B. Arthur, “Inductive reasoning and bounded rationality,” American Economic Review (Papers and Proceedings), vol. 84, pp. 406–411, 1994.
3. W. Banzhaf, “Self-replicating sequences of binary numbers — the build-up of complexity,” Complex Systems, vol. 8, pp. 205–215, 1994.
4. M. A. Bedau and C. T. Brown, “Visualizing evolutionary activity of genotypes,” Artificial Life, vol. 5, no. 1, pp. 17–35, 1999.
5. M. A. Bedau and N. H. Packard, “Measurement of evolutionary activity, teleology, and life,” in Proceedings of Artificial Life II, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, (eds.), Addison-Wesley: Redwood City, CA, 1991, pp. 431–461.
6. M. A. Bedau, E. Snyder, C. T. Brown, and N. H. Packard, “A comparison of evolutionary activity in artificial evolving systems and the biosphere,” in Proceedings of the Fourth European Conference on Artificial Life (ECAL97), Brighton, P. Husbands and I. Harvey, (eds.), MIT Press: Cambridge, MA, 1997, pp. 125–134.
7. M. A. Bedau, E. Snyder, and N. H. Packard, “A classification of long-term evolutionary dynamics,” in Proceedings of Artificial Life VI, Los Angeles, C. Adami, R. Belew, H. Kitano, and C. Taylor, (eds.), MIT Press: Cambridge, MA, 1998, pp. 228–237.
8. E. J. W. Boers and H. Kuiper, “Biological metaphors and the design of modular artificial neural networks,” Master’s thesis, Departments of Computer Science and Experimental Psychology, Leiden University, The Netherlands, 1992.
9. S. Bullock and M. A. Bedau, “Exploring the dynamics of adaptation with evolutionary activity plots,” Artificial Life, vol. 12, no. 2, pp. 193–197, 2006.
10. A. D. Channon, “Passing the ALife test: Activity statistics classify evolution in Geb as unbounded,” in Advances in Artificial Life: Proceedings of the Sixth European Conference on Artificial Life (ECAL2001), Prague, J. Kelemen and P. Sosik, (eds.), Springer-Verlag: Heidelberg, 2001, pp. 417–426.
11. A. D. Channon, “Improving and still passing the ALife test: Component-normalised activity statistics classify evolution in Geb as unbounded,” in Proceedings of Artificial Life VIII, Sydney, R. K. Standish, M. A. Bedau, and H. A. Abbass, (eds.), MIT Press: Cambridge, MA, 2003, pp. 173–181.
12. A. D. Channon and R. I. Damper, “Perpetuating evolutionary emergence,” in From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB98), Zurich, R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. Wilson, (eds.), MIT Press: Cambridge, MA, 1998, pp. 534–539.
13. A. D. Channon and R. I. Damper, “Towards the evolutionary emergence of increasingly complex advantageous behaviours,” International Journal of Systems Science, vol. 31, no. 7, pp. 843–860, 2000.
14. D. Cliff, I. Harvey, and P. Husbands, “Incremental evolution of neural network architectures for adaptive behaviour,” University of Sussex School of Cognitive and Computing Sciences, Tech. Rep. CSRP256, 1992.
15. P. Dittrich and W. Banzhaf, “Self-evolution in a constructive binary string system,” Artificial Life, vol. 4, no. 2, pp. 203–220, 1998.
16. I. Harvey, P. Husbands, and D. Cliff, “Issues in evolutionary robotics,” in From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92), J. A. Meyer, H. Roitblat, and S. Wilson, (eds.), MIT Press/Bradford Books: Cambridge, MA, 1992, pp. 364–373, also available as report CSRP219, School of Cognitive and Computing Sciences, University of Sussex.
17. J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press: Ann Arbor, MI, 1975, 2nd edition: MIT Press, 1992.
18. P. Husbands, I. Harvey, and D. Cliff, “Analysing recurrent dynamical networks evolved for robot control,” in Proceedings of the Third IEE International Conference on Artificial Neural Networks (IEE-ANN93), IEE Press: London, 1993, pp. 158–162.
19. W. B. Langdon, “Pfeiffer – A distributed open-ended evolutionary system,” in Proceedings of the Joint Symposium on Socially Inspired Computing, B. Edmonds, N. Gilbert, S. Gustafson, D. Hales, and N. Krasnogor, (eds.), AISB: The Society for the Study of Artificial Intelligence and the Simulation of Behaviour: Brighton, 2005, pp. 7–13.

20. K. Lindgren, “Evolutionary phenomena in simple dynamics,” in Proceedings of Artificial Life II, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, (eds.), Addison-Wesley: Redwood City, CA, 1991, pp. 295–312.
21. C. C. Maley, “Four steps toward open-ended evolution,” in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99), W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, (eds.), Morgan Kaufmann: San Francisco, CA, 1999, pp. 1336–1343, volume 2.
22. W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” Bulletin of Mathematical Biophysics, vol. 5, pp. 115–133, 1943.
23. N. H. Packard, “Intrinsic adaptation in a simple model for evolution,” in Santa Fe Institute Studies in the Sciences of Complexity, Vol VI: Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (Artificial Life I), C. G. Langton, (ed.), Addison-Wesley: Redwood City, CA, 1989, pp. 141–155.
24. K. Sims, “Evolving 3d morphology and behavior by competition,” in Proceedings of Artificial Life IV, R. A. Brooks and P. Maes, (eds.), MIT Press: Cambridge, MA, 1994, pp. 28–39.
25. K. Sims, “Evolving virtual creatures,” in Proceedings of Computer Graphics (Siggraph ’94) Annual Conference Proceedings, ACM Siggraph: New York, 1994, pp. 15–22.
26. R. K. Standish, “An ecolab perspective on the Bedau evolutionary statistics,” in Proceedings of Artificial Life VII, M. A. Bedau, J. S. McCaskill, N. H. Packard, and S. Rasmussen, (eds.), MIT Press: Cambridge, MA, 2000, pp. 238–242.
27. A. Stout and L. Spector, “Validation of evolutionary activity metrics for long-term evolutionary dynamics,” in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005), H.-G. Beyer and U.-M. O’Reilly, (eds.), ACM Press: New York, 2005, pp. 137–142, volume 1.
28. T. Taylor and J. Hallam, “Replaying the tape: An investigation into the role of contingency in evolution,” in Proceedings of Artificial Life VI, Los Angeles, C. Adami, R. Belew, H. Kitano, and C. Taylor, (eds.), MIT Press: Cambridge, MA, 1998, pp. 256–265.
29. T. Taylor and C. Massey, “Recent developments in the evolution of morphologies and controllers for physically simulated creatures,” Artificial Life, vol. 7, no. 1, pp. 77–87, 2001.