Complex Systems **5** (1991) 279–298

Evolution of Bit Strings: Some Preliminary Results[*]

Harald Freund[†]
Robert Wolter
Physics Department, University of Wuppertal,
Gauss-Straße 20, D-5600 Wuppertal 1, Germany

Abstract. In this paper we show some preliminary results of simulations with a population of bit strings. We present the ideas of our model and point out the difficulties in implementing them. Although an open evolution is one of the final aims, we can present some interesting results with finite and even very small systems. One important result is the evidence that introducing startcodons in the decoding scheme is reasonable. This seems to have biological relevance since startcodons are used in natural decoding processes.

1. Introduction

1.1 General overview

During the last few years, attempts to model systems with evolutionary properties have become more and more numerous. The fundamental question one wants to answer is essentially: *What are the basic and necessary ingredients that make evolution from simple building blocks to large and complex structures possible, likely, or even inevitable?*

The whole evolution from single-cell bacteria to human beings depends on a very elaborate mechanism for self-reproduction. But it is generally believed now that even the first self-reproducing molecules had a rather complicated structure—they were RNA-like—which made their appearance by chance rather improbable. So there remains a gap between a primordial soup and the first self-reproducing molecules, and even more so between these and primitive organisms like bacteria. There are various approaches trying to solve this puzzle. They include Eigen's hypercycle [1], autocatalytic networks [2], the Coreworld simulations of Rasmussen et al. [3], and Fontana's AlChemy [4].

[*]Part of this work was done while participating in the Institute for Scientific Interchange workshop on "Complexity and Evolution."

[†]Electronic mail address: `freund@physik.uni-wuppertal.dbp.de`

The guiding principle in specifying our model was that it should be as simple as possible, yet incorporate some features that we consider essential:

1. Individuals should be built up from several building blocks in order to provide a huge state space due to combinatorial explosion.
2. There should be no frequency-independent fitness, that is, no absolute goal.
3. Selection should play at least partially on individual building blocks in order to have "mild" mutations, where changes in one block leave most of the score of an individual unchanged.
4. We want to model a discrete stochastic system, not a system of differential equations. In an infinite system noise could be suppressed by the law of large numbers, and we could have limit cycles or chaotic attractors. In our finite systems we must approach some invariant distribution, though this might be reached so extremely slowly that it may be irrelevant.

1.2 Our model

We chose a population of fixed-length bit strings—they can be viewed as DNA-like molecules having only two instead of four bases or some very small molecules directly decoded by a bit string—swimming in a well stirred soup. This way interactions between two of them can be simulated by taking them at random with equal probability. For simplicity, these bit strings will be called "animals" in the following. For each animal we define a real variable storing the score (or fitness) of that animal. We should stress here that this "score" or "fitness" is *not* a prespecified *global* function, because the fitness of each animal is determined by its interactions with *randomly* chosen others. So there is only a "population-dependent" fitness that continually changes as the population changes. The present population, considered as a point in the state space of all possible populations, is the local environment for each animal in which its local fitness is evaluated in a prespecified way.

The following two elements are necessary to create a sequence of generations:

- An interaction scheme that changes the scores of the animals taking part in the interaction; details are given in section 2.1.
- An updating or reproduction scheme that uses the scores to decide which animals go on to the next generation and which drop out and are replaced by new ones. The reproduction involves errors produced by mutations; details are given in section 2.2.

Implementing these two points leads to problems that are of either a technical or a conceptual nature. Let us explain what we mean by *technical* versus *conceptual.*

Mutation that can be implemented in different ways is used to add new material to the population during the reproduction phase. We apply single-point mutation, that is, "single-bit flip" (except in section 5 where we add "cut and splice" [5]), when an animal produces "offspring." This is simple to implement because it takes only one random number. There are other possibilities that cause on average the same mutation rate [6]. But as the main feature of mutation is to introduce new material, the results do not depend much on how it is done. We thus claim that how mutation is implemented is only a technical problem.

Whether to use mutation as a source of variability or other genetic operators such as inversion, replication, or the parasite model [7], is more a conceptual problem. The same is valid for sexual reproduction, which can employ crossover, assortive mating, recombination [8], and cut and splice, as described in section 5.3.

The decoding scheme that is presented in detail in section 3 is also a conceptual problem. If decoding is done in an unsuited way our final goal—a "true" evolution—can never begin.

We should point out, however, that for many features of an evolving system one cannot decide right from the start whether an implemental alternative is merely a technical or a conceptual problem.

2. Interaction and reproduction

2.1 The interaction scheme

Each time step of the simulation consists of two parts: first an interaction phase that calculates the scores of all animals, and thereafter a reproduction phase as described in section 2.2. The interaction phase first sets all scores to zero and then executes a number of "basic interactions." A basic interaction takes two animals i and k purely at random, selects one gene in each of them (g_i and g_k)—again at random—and performs the following update of the scores s_i and s_k:

$$\begin{aligned} s_i &\leftarrow s_i + A(g_k, g_i) \\ s_k &\leftarrow s_k + A(g_i, g_k) \end{aligned} \tag{2.1}$$

where A denotes a random matrix with values taken from the interval $[-1, +1]$. The number of basic interactions was typically chosen to be a multiple, called α, of the number of animals. Most simulations were done with $\alpha = 10$. Since two animals participate in one basic interaction, each animal takes part in 20 interactions on average. But we also conducted simulations with $\alpha = 20$, 30, and 40.

2.2 Reproduction

In all simulations we implemented synchronous reproduction throughout the population. Let us consider some alternatives:

1. One simple reproduction scheme calculates the average score and removes all animals that are below average and replaces them by mutating survivors. Notice that this implies the somewhat "unbiological" feature that well adapted animals do not die, but survive without being changed by mutations. But this mechanism is easy to implement, needs no sorting, and is carried out in time $O(n)$ where n is the number of animals in the simulation.

2. Another scheme we implemented replaces a *constant fraction* of the population by mutants after every time step (i.e., in every reproduction phase). We chose the worst quarter of the population to be replaced. The new animals are mutants from

   (a) the best animals according to score (we choose from the best quarter),

   (b) all surviving animals, and

   (c) all animals, even from those that will be replaced.

One argument for case (c) is the fact that even a mutant that is at first unsuccessful could contain some genetic material that proves useful in the future ("hopeful monsters"). Nevertheless we favored (a) because we have another possibility to preserve neutral genetic material (see section 3.3) and we wanted to simulate a fairly small population (64 or 128 individuals) for reasons of computational economy.

Another aspect of our model is the probability $p(s_i)$ for an animal i with score s_i to be chosen for reproduction. Very often a Fermi distribution of the form

$$p(s_i) = \frac{\text{constant}}{1 + \exp[(s_i - \bar{s})/C]} \tag{2.2}$$

is taken. Here, $\bar{s} = (1/N) \cdot \sum_{i=1}^{N} s_i$ denotes the average score and C is a temperature-like parameter. We used $C = 0$, which chooses all animals above average with equal probability. In addition we replaced $\bar{s}$ by $\tilde{s}$, where $\tilde{s}$ denotes a value that is smaller than the highest quarter of all scores but strictly larger than the rest (corresponding to case (a) above). We favored this case because it lets us use simple, equally distributed random numbers. Another advantage of this implementation is that we can easily estimate the number of animals n_t that die after a certain number of time steps t if the animals die purely at random. Explicit calculations are given in appendix A.

3. Decoding the bit strings

We subdivided the bit strings into functional groups that we call "genes." The name genes was chosen for convenience and should not be taken too literally. The concept of genes is closely connected with the distinction of genotype and phenotype, as sketched in figure 1. The genotype is the set of

$$\begin{tikzcd}
\text{genotype (set of genes)} \arrow[r, "\text{development}"'] & \text{phenotype (character of animal)} \arrow[r, "\text{fitness}"'] & \text{populations (interaction)} \arrow[ll, bend left]
\end{tikzcd}$$

Figure 1: Genotype-phenotype relation.

genes stored in the DNA. This determines the phenotype, which in turn determines the fitness. The fitness we can ascribe to an animal is not evaluated by looking at the genotype; it follows indirectly via the phenotype.

As a consequence the genotype-fitness relation is very nonlinear and unpredictable. In our model we have no explicit distinction between genotype and phenotype. The genes can interact directly via the interaction matrix A as described in section 2.1. This matrix implicitly contains the gene-phenotype map. The randomness of the values of the matrix elements is supposed to model the complexity and unpredictability of the gene $\rightarrow$ phenotype $\rightarrow$ fitness map.

The genes are decoded using a binary tree. To read a bit string from any prescribed position forward, one starts at the root and moves down the tree while the bits in the animal decide which way to go—**0** turns left, **1** turns right—and stops if a leaf (which contains the number of the gene) is encountered. This procedure is continued until the whole bit string of an animal is decoded. In most simulations we worked with a randomly generated tree that is shown in table 1 and in figure 2. It decodes 20 genes with lengths between 3 and 8 bits.

The main reason for using a binary tree instead of a randomly generated list of genes (that can also be listed in a table like table 1) is that this way we are able to decode every possible bit string without ambiguity. Each path through the binary tree leads to a gene after at most 8 left-right decisions. There is neither arbitrariness (one sequence $\rightarrow$ many interpretations) nor uninterpretable sequences. In addition this tree allows a simple extension of the model where the tree is able to grow: A leaf can be replaced by a node that adds two genes to the tree in the next level, so new and longer genes can be introduced and effectively change the environment the animals are living in. But don't forget that even without a growing tree we have *no static* environment.

Even with a given representation there are different ways to decode a bit string. We describe some of them in detail in the following subsections.

3.1 Genes packed in highest density

The largest number of functional groups in a bit string can be obtained by joining the beginning and the end of the animal, creating a circular

Gene	Length	Bit representation
A	3	001
B	3	011
C	3	110
D	3	111
E	4	0000
F	4	0100
G	4	0101
H	4	1000
I	4	1010
J	4	1011
K	5	10011
L	6	000100
M	6	000101
N	6	000110
O	6	000111
P	6	100100
Q	8	10010100
R	8	10010101
S	8	10010110
T	8	10010111
sc	3	110

Table 1: Bit representation of the genes and of the startcodon (see section 3.3) that were used in all simulations. The startcodon is denoted by "sc."

sequence. A gene starts at every position; that is, an animal with 32 bits stores 32 genes. In this case the genes overlap maximally and we see strong correlations between genes. So the interaction between animals is not the sum of independent interactions between single genes, but of interactions between groups of genes. This is also known as the hitchhiker effect. For a detailed example see figure 3. Simulation results are presented in section 5.1.

3.2 Gene beside gene

(a) The next possibility avoids overlaps of functional groups by scanning the bit string from the beginning to the end and starting a new gene just after the previous one is completed. The last bits, which belong to no gene, are not interpreted. An example is shown in figure 4.

A point mutation at the end of an animal, decoded this way, will have a small affect on the genetic contents. The same is true for a mutation at the beginning of the string *if* this does not change the reading frame. But

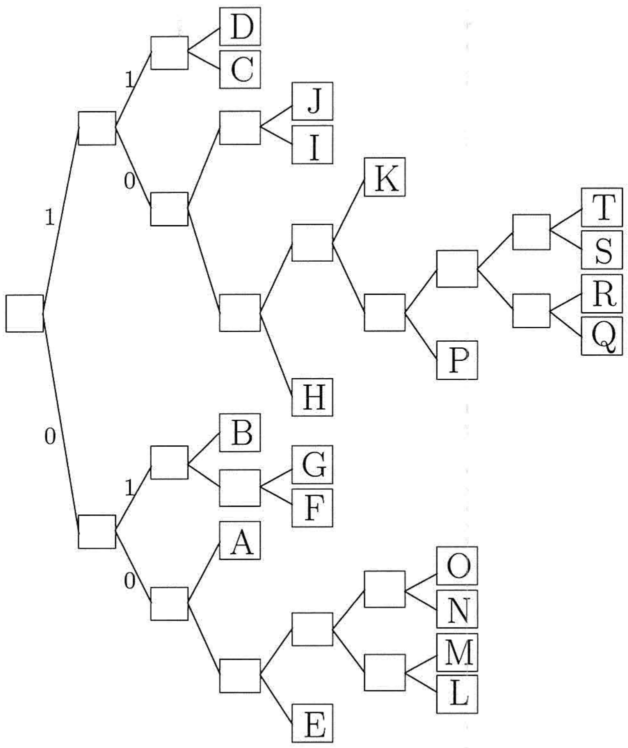

Figure 2: Bit representation of the genes that were used in all simulations, shown as a tree (as mentioned in section 3).

Bits						Gene
0	0	0	1	0	0	L
	0	0	1			A
		0	1	0	0	F

Figure 3: Bit string decoded as described in section 3.1. In this example the genes start at each position in the bit string so the density of genes is maximal. The substring "000100" is always decoded as the sequence "LAF," which is followed by a gene that starts with "100" (compare table 1 and figure 2).

```
  C  |        S          |    F    |   D   |    G     |      N        |   A  | -
1 1 0| 1 0 0 1  0  1 1  0| 0 1  0 0| 1 1 1 | 0 1  0 1 | 0 0 0  1 1 0  | 0 0 1| 0
 ↓
1 0 0  1 0 0| 1  0  1 1| 0 0 1 | 0 0 1| 1 1 0| 1  0 1 0| 0 0  1| 1 0  0 0| 1 0
       P    |     J    |   A   |   A  |   C  |    I   |   A   |    H    | - -
```

Figure 4: Example of a single mutation at bit number 2 that completely changes the reading frame. The genes are labeled as in table 1 and figure 2.

a mutation in the first gene that converts it to a gene of different length completely changes the contents by altering the reading frame (compare figure 4). In this way point mutation can cause big leaps in the space of all possible animals, and improvements can only result from "jumps" right onto an improved point. In contrast we want mutations to be a minor disturbance (or a kind of low amplitude noise) that changes only a small amount of information and allows for gradual improvement. Or, to put it another way, we want point mutation to be a local operator.

(b) This locality can be restored by cutting out the gene that is to be mutated and putting it at the end of the animal, so each point mutation results in only minor changes. Thus it seems that we can avoid the nonlocality of mutations, but only by means of a trick that does not seem to be very natural.

3.3 Introducing startcodons

In a third approach the bit string is scanned for appearances of a special bit sequence called a "startcodon," which signals the beginning of a gene.

On the one hand there can still be overlapping genes if a substring of a gene is equal to the startcodon. In the realization shown in table 1 and figure 2, for example, we chose the sequence "110" (equal to gene C in our binary decoding tree) as the startcodon. By this choice no two genes can overlap.

On the other hand there are parts of an animal—called "junk" from now on—that carry no information because there is no startcodon. This could pose a problem if animals without any startcodon could dominate. But, as these do not interact, their score remains zero. Since the average score of the successful animals is positive, these animals are killed in the long run.

This decoding scheme defines mutation as a local operator because it can affect at most two genes (if a startcodon overlapping with a gene is altered, compare table 1). Let us explain this with an example. Figure 5 shows the same bit sequence that was presented in figure 4. Startcodons are denoted by "sc" and junk by "-". If we now perform the same mutation as in figure 4, only a minor change results. Two genes would be changed if the mutated bit were from the second startcodon.

```
  s c  |       S       |          -| s c  |  I  |- -| s c  |  A |-
               | s c   |    F    |
1 1 0|1001  0|1  1  0|01   00|1|1  1  0|10 10|00|1  1  0|0 0 1|0
  ↓
1 0 0 1001  0|1  1  0|01   00|1|1  1  0|10 10|00|1  1  0|0 0 1|0
- - - ----  -| s c   |    F    |-| s c  |  I  |- -| s c  |  A |-
```

Figure 5: Example of a single mutation at bit number 2 that changes only slightly the contents of the animal because startcodons are used. The genes are labeled as in table 1 and figure 2.

We see that startcodons are an effective method of guaranteeing that mutation is a local operator, so we should not be surprised to find startcodons being used in real DNA sequences.

The second positive feature of startcodons is the introduction of junk. Junk can be a source for sudden change because it can accumulate changes that are not interpreted. When a startcodon is created by a mutation in a junk region, new information is evoked. But on average the effective rate of mutation is lowered by the presence of junk since we used only a single point mutation per new created animal. (This would not be true if we used a fixed mutation rate per bit).

4. The role of randomness

There are three aspects of randomness in our model:

1. the interaction matrix $A(g, g')$,
2. the choice of the interacting animals, and
3. the choice of interacting genes in the animals.

A possible objection against our model could be that the whole system might have too much randomness in it; in particular the third aspect could be made deterministic by defining some genes as "active" in the bit string and using only these for the score evaluation. Alternatively, for example, we could implement a "complete" interaction; that is, each gene of the first animal could interact with each gene of the second animal. We played with different implementations, but the main features of the system stayed the same.

We now want to give some arguments in favor of the sources of randomness mentioned above. The first concerns the randomness of the interaction matrix. In real nature, the interaction between genes is an extremely complicated process, involving the folding of the DNA itself, the folding of the expressed protein, its effect on the phenotype, and so forth. All these steps are most likely very sensitive to small details, and we consider it neither feasible nor desirable to model such a detailed dynamics. Instead of an explicit phenotype/genotype distinction and a deterministic prescription for the interaction between genes, we argue that a random matrix as in our model accounts best for what can happen in "reality."

The answer to the second point was also mentioned earlier in this paper. We consider the environment in our model as a well-stirred fluid where the animals are floating, so an interaction between two of them is simulated best by picking them at random.

Let us now come to the third point. If we think of the bit strings as real one-dimensional strings, we cannot expect them to line up from head to tail when colliding. It is more likely that only a small part of each is in contact with the other, so we can equate this part with one gene. If we consider the bit string as a three-dimensional structure the argument is similar: only parts on the surface of this structure are active, and again a collision results in just two genes interacting. Finally, if the interaction passes through some reading machinery as in real biological systems, then the effect of this machinery will be highly complex, so again a random model seems most appropriate.

So it seems fairly reasonable to work with all those random elements we described before.

5. Computational results

5.1 Genes packed in highest density

Under the decoding scheme described in section 3.1 a population evolves to a kind of (statistically) stationary state after a rather short time. It is essentially independent of the initial population and of the interaction rate α as defined in section 2.1. The statistics in this state depend on the interaction matrix A, which can be seen by contrasting it with a random population. Genes of the same length (e.g., A to D or L to P, see table 1) would occur with the same probability and longer genes are suppressed. In our system the gene frequencies soon reach a state with constant averages but strong fluctuations around these mean values. The strong fluctuations are caused by the replacement of a quarter of the population each time step. When reaching this state, there are still many different "species" in the population, where a species is defined by its gene content but neglects the order of these genes (and any junk, see section 5.3). In contrast to the random case, genes of the same length do not have the same frequencies. Even an enhancement of a longer gene compared to a short one is possible. For example, in a random population gene L is suppressed by a factor of 2 relative to gene K; in our stationary state, gene K is suppressed by a factor of 2.5 relative to gene L.

We blame this appearance of a stationary state on the strong correlations shown in figure 3. Correlations are less pronounced when we start a gene at every second bit position. Although this changes the kind of stationary state, the system still reaches a stationary state. In contrast to the case where a gene started at each bit, it can be described as one "quasispecies" consisting mainly of one successful group of genes (for a complete definition of "quasispecies" see [1]). The whole population consists of the optimal species and of animals that have a small Hamming distance from this optimum, the latter corresponding to recent mutation. This behavior was also robust against changes in the interaction rate.

Because this decoding scheme lacks the ability to change, we did not continue using it.

5.2 Genes beside genes

The next mechanism we implemented was the decoding scheme described in section 3.2. In most simulations we chose variant (b), where the mutated gene is cut out and put at the end of the animal. Though this is a somewhat artificial trick to make mutation local, systems simulated this way showed promising results.

During a simulation one gene can be absent for a long interval, but may later reappear and dominate (compare gene O in figure 6). So a continually changing environment is possible, which implies there is no frequency-independent fitness function that can be evaluated externally. In particular there may be cooperation between genes (genes H and O in figure 6) and mutual suppression (genes F and B in figure 7) that could not be obtained from a global fitness function.

The characteristic behavior of our system—bistability and genetic takeover—is independent of the initial population and independent of the random matrix chosen. But it strongly depends on the value of the interaction parameter α (as defined in section 2.1). Higher values of α result in long-living metastable states or "punctuated equilibria" (compare figure 6) where no recurrence of an earlier state is seen. The number of possible stationary states depends on the interaction matrix. Lower values of α lead to faster decay of such states, where the same state recurs frequently (compare figure 7). There are longer phases where genes A and B dominate, alternating with phases where genes F and H lead.

Figures 6, 7, 10, and 11 show the frequencies for several genes of the binary tree presented in figure 2 for different simulations with 64 individuals each. The abscissae describe the simulation time steps and the ordinates indicate the gene frequencies. Note that the ordinates show different scales.

Figures 6 and 7 result from runs that used the decoding scheme described in section 3.2, with the same interaction matrix and the same initial population; they differ only in the interaction parameter α. Each animal consists of 32 bits. We see a rich behavior with at least three metastable states (genes A and B, genes F and H, and genes H and O).

To sum up, like natural evolving systems our system shows stability on the one hand and the possibility to change on the other. This is so in spite of our assumption of spatial homogeneity, which prevents the formation of different niches and further suppresses the formation of structure.

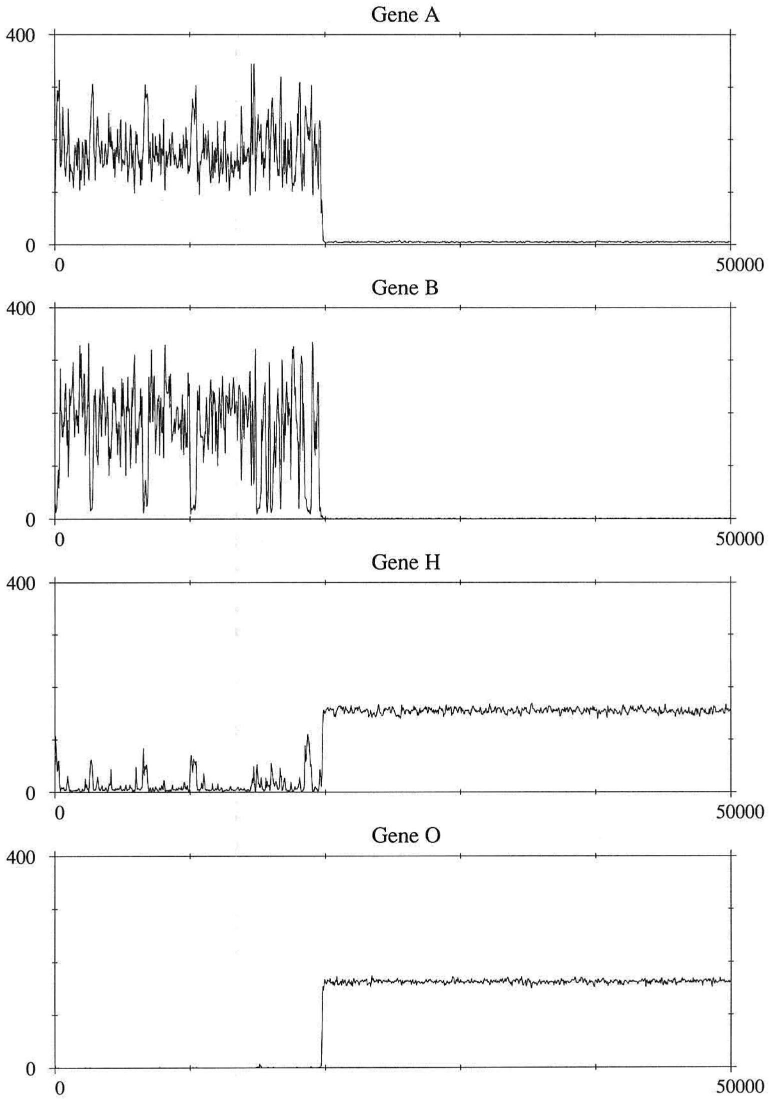

Figure 6: Example for a "genetic takeover." In the beginning two genes, A and B dominate for a rather long time but then two different genes, H and O, turn up and take over. We do not show genes with small frequencies; for example, H and O are accompanied by N, but its amplitude is below 50. Here and in the following figures only the most successful genes are shown. Typically at least 12 of the 20 possible genes have negligible frequencies. The multiple α defined in section 2.1 is 20.

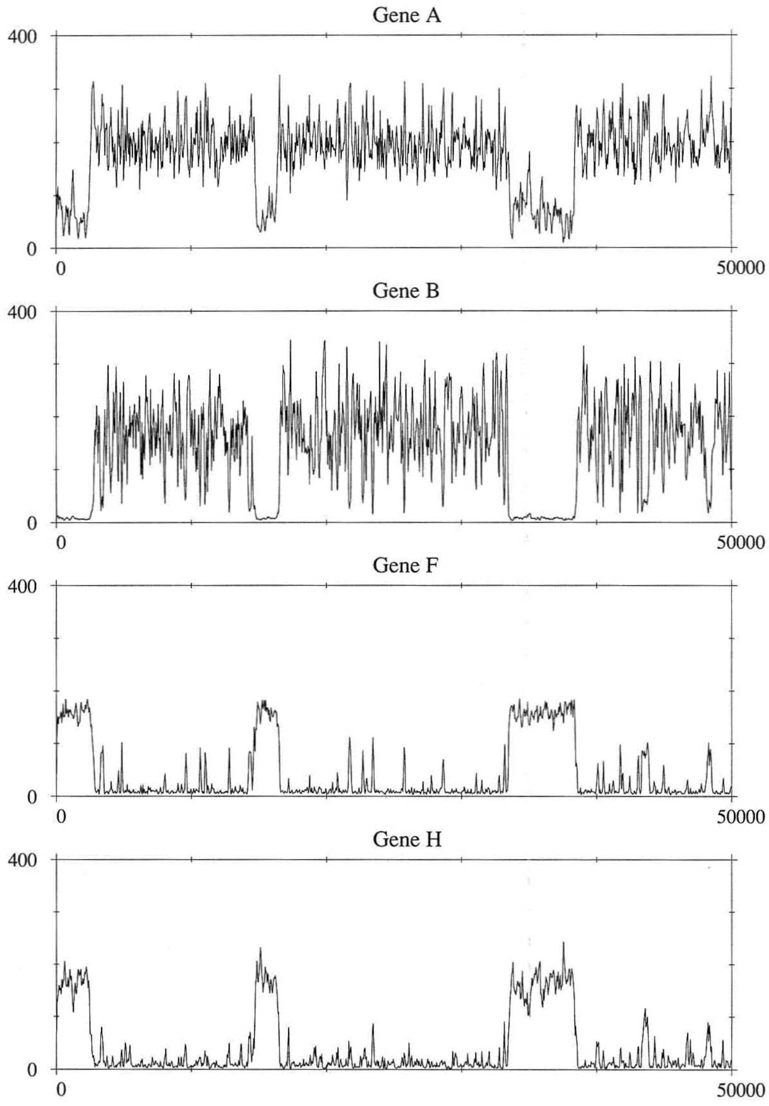

Figure 7: Bistability of the gene frequencies. One stable state is characterized by the dominance of the genes A and B, the other one is dominated by genes F and H. The same initial population as in figure 6 was used, but with $\alpha = 10$.

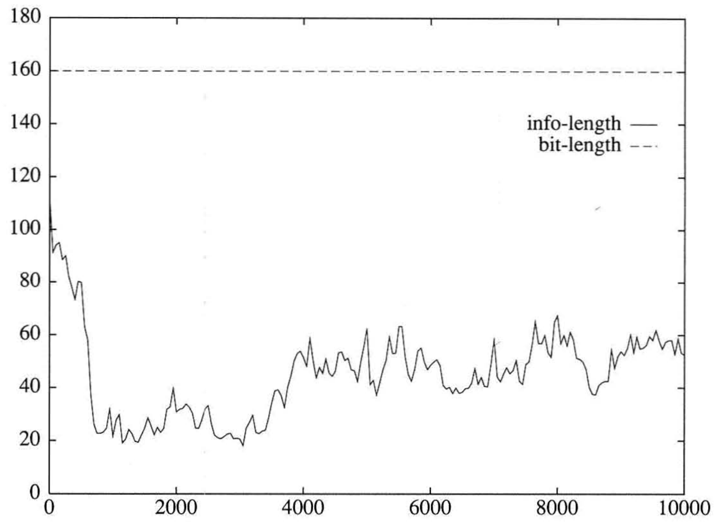

Figure 8: This plot compares the bit length (which was held constant) to the info-length as defined in section 5.3. Except for the initial 700 time steps, the info-length stays below 60 bits, so only 38% of an average animal's bits are decoded as information. This way the mutation rate is effectively decreased.

5.3 Using startcodons

The use of startcodons leads to systems with properties similar to those we described in section 5.2. We could present plots similar to figures 6 and 7, but this is not very instructive. Due to the presence of junk, animals of 32 bits now have fewer genes (compare figures 4 and 5). So we increased the length to 64 (and later up to 160) bits and introduced what we call the information-length or, for short, "info-length." The info-length is the number of bits decoded as information (i.e., belonging either to a startcodon or to a gene). The animal shown in figure 5 has an info-length of 28 bits before and 20 bits after the mutation. When measured this way, we see that the lengths of the animals can change.

An average random population of animals of 64 bits has an average info-length of 40 to 45 bits; that is, two-thirds of the animals are decoded as information and one-third as junk. The info-length drops to 20 to 25 bits very soon (this leads to animals with at most 5 genes). What are the reasons for this behavior? As described in section 2.2, one-fourth of the population is replaced each time step by 1-bit-flip mutants of the highest ranking quarter. Thus there is a rather high mutation rate, which is effectively decreased when animals have large regions of junk where a bit flip usually leads to no change

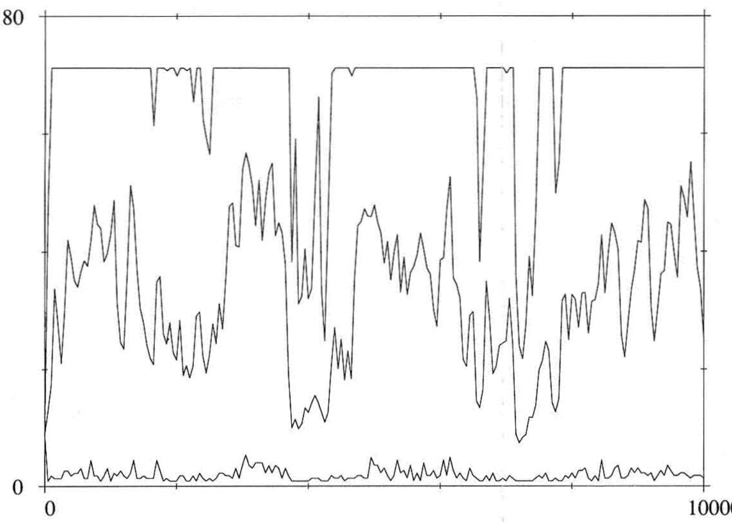

Figure 9: The three curves present (from top to bottom) the maximal bit length of animals (which could be 160 in this run), the average bit length, and the minimal bit length (which could be at least 2). The average value has strong fluctuations about the mean value of approximately 80 bits.

(because no startcodon is produced). This is reminiscent of Holland's schema theorem [9].

To further examine the decreasing info-length we increased the bit length of the animals to 160 bits. A random initial population has an average info-length of 110 to 120 bits; that is, more than two-thirds of an animal are decoded as information. Figure 8 shows a typical simulation result. At first the info-length drops to approximately 20 bits and later increases to 40 to 60 bits. Thus the relative info-length (i.e., info-length normalized by the bit length) is 0.25 to 0.38 both for short (64 bits) and for long (160 bits) animals.

To get deeper insight into this phenomenon we introduced a second genetic operator, "cut and splice" [5]. Cut and splice takes two animals, cuts each at a random position—not necessarily the same position—and builds two new animals, each consisting of two parts, one from each ancestor. This way length changes are possible. We allow animals of a length between 2 and 160 bits, starting with animals all having the same length. We still find punctuated equilibria where some gene(s) dominates the population. During some periods longer animals prevail, and during others short or medium sized animals (see figure 9). Figures 10 and 11 show results for a system using the same interaction matrix as the one used to produce figures 6 and 7. Different initializations lead to different sequences of metastable states. In addition to the three metastable states we described in section 5.2, we now see outbursts of gene P preceeding either the newfound stable state with high frequencies of genes J and M, when the initial bit length is 20, or the previous stable state of genes H and O, when the initial bit length is 60. But there is no trend concerning the bit length of the animals.

Since we only conducted simulations lasting 10,000 to 50,000 time steps we are never sure that there could not be a change of state in even longer runs, although some populations seem to settle into what looks like a "global optimum" (compare figure 10).

In this paper we only used the gene frequencies to discuss the encountered phenomena. But, in addition, we also printed out successful animals (i.e., those animals that stayed in the simulation for more than 20 to 25 time steps). We found that those animals contained only the successful genes and could be grouped together in "species." In the simulation leading to figure 10 we saw mainly the following species: "O," "H," "HO," and "OH." It is interesting to explore these species, to look at their age distribution and their relative strength (i.e., number of animals belonging to this species). But as this goes beyond the scope of this work we have to defer to a future paper.

6. Conclusion

Our model for evolution of bit strings showed some general properties that are summarized here.

When we compare the theoretical numbers of individuals reaching a specified age with our results, we see directly the effect of selection: it leads to a deviation from an exponential age distribution due to successful animals becoming older and unfit mutants dying very soon (see also appendix A). This is independent of the decoding scheme.

When we decode the bit sequences without using startcodons, we see that our system tends to develop into (meta)stable limit states. There is bistability if the interaction parameter is rather small; whereas a stationary state takes over when this parameter is increased. If startcodons are implemented the dynamics more likely shows multistability without a defined limit state. We still see some of the states that appeared in the system when using different decoding schemes.

Another positive feature of introducing startcodons is the presence of junk. Thereby the system is able to compensate for the selection pressure caused by the relatively large mutation rate by using only a fraction of the bit string as information. This leads to a lower number of genes in the animals and also implies the possibility of sudden change that ends the dominance of successful genes.

The models we implemented do not really lead to an open evolution, but they also do not get stuck in a local maximum of fitness without having a chance of further development.

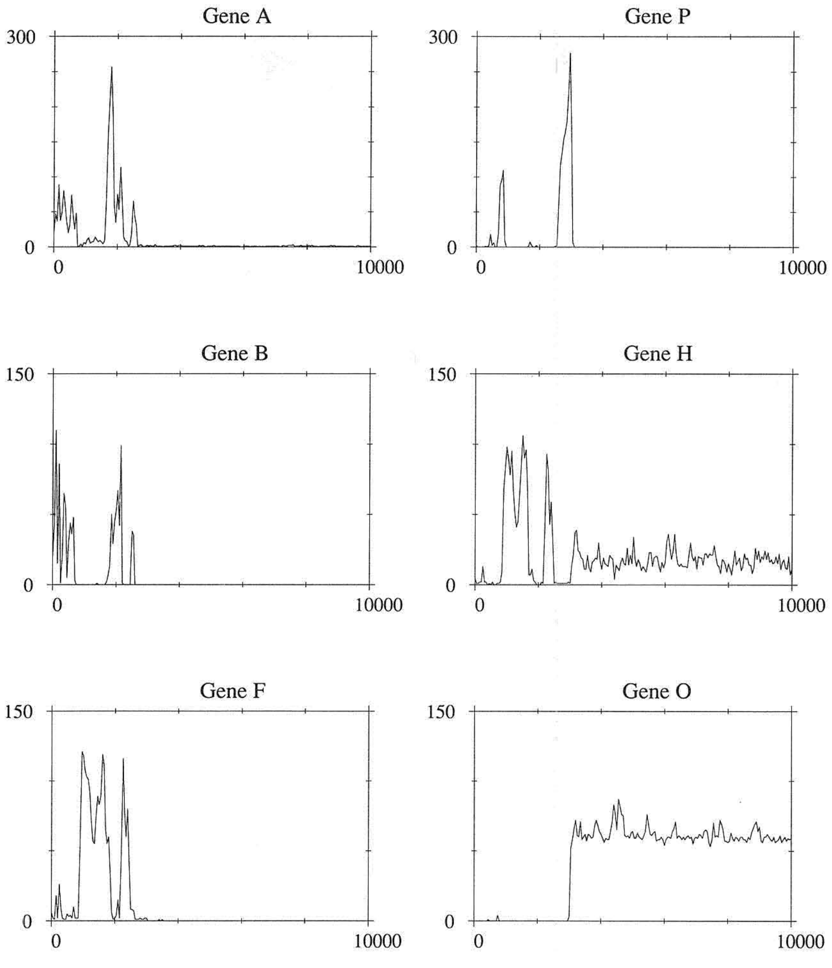

Figure 10: Figures 10 and 11 result from runs that used the decoding scheme described in section 3.3 and employed crossover to allow changes in bit length (see section 5). They illustrate the sensitive dependence on the initial bit length of the start-population, that is, the lack of a unique optimal state.

This figure shows the result of a simulation with an initial population of bit length 20 and a maximal bit length of 160. Notice that, in the start phase with high fluctuations of all frequencies, gene P shows strong outbursts, but it vanishes after a very short time of domination. After its second burst a (meta)stable state with genes H and O emerges.

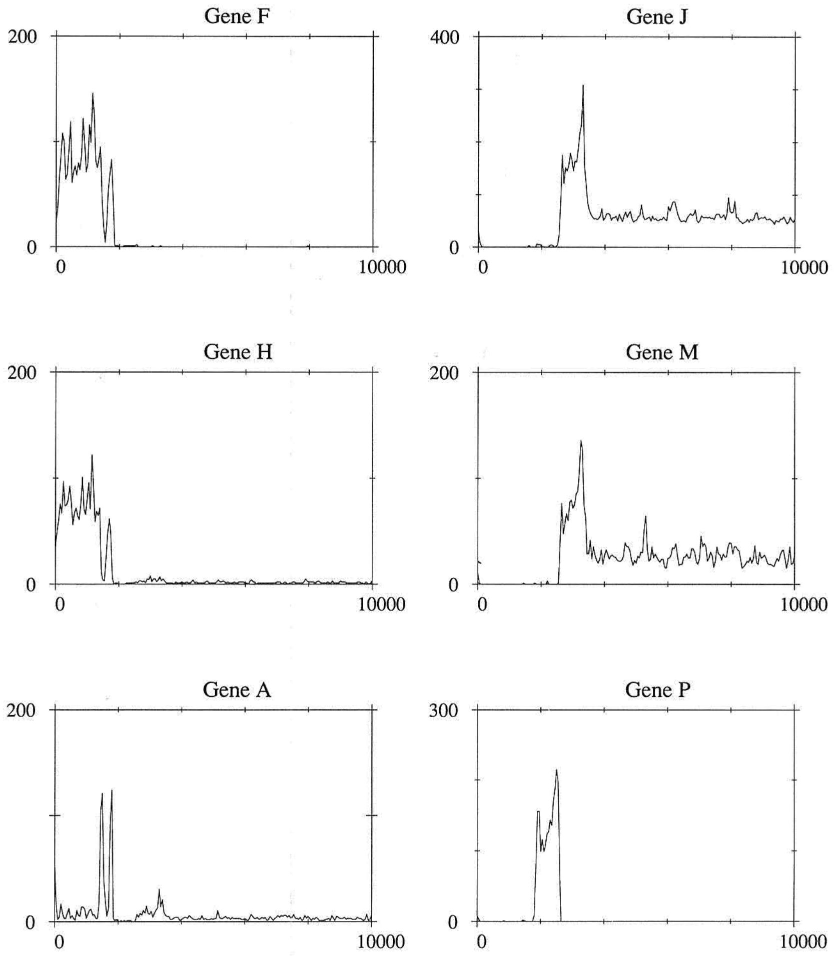

Figure 11: Results of a simulation with the same parameters as the one described in figure 10, but the population started with a bit length of 60. It shows similar results, but in this case the genes J and M take over.

When we consider only the gene frequencies, we do not get all the important information about a population. We also have to take into account "species" to get a complete picture of the dynamics. This will be part of future work.

Appendix A

We now derive results for a simulation where animals are killed in a purely random way. We calculate the number of animals n_t that die at age t for a simulation of N animals that lasts T time steps, and where in each time step a part $p \in [0, 1]$ of the animals is killed. The total number of killed animals

$$\sum_{t=1}^{\infty} n_t = p \cdot N \cdot T \tag{A.1}$$

considering the recursive relation

$$n_{t+1} = (1 - p) \cdot n_t \tag{A.2}$$

leads to

$$\begin{aligned} n_1 &= p^2 \cdot N \cdot T \\ \textstyle\sum_{t=k}^{\infty} n_t &= (1-p)^{k-1} \cdot p \cdot N \cdot T. \end{aligned} \tag{A.3}$$

The highest age in the population is on average given by the lowest value τ that satisfies $\sum_{t \geq \tau} n_t < 1.0$:

$$\tau = \left\lceil 1 - \frac{\ln(p \cdot N \cdot T)}{\ln(1-p)} \right\rceil . \tag{A.4}$$

$\lceil a \rceil$ denotes the smallest integer larger than a. A usual simulation conducted with the values $p = 0.25$, $N = 64$, and $T = 10{,}000$ results in $n_1 = 40{,}000$ and $\tau = 43$. From this we conclude that if all animals were killed purely at random the oldest ones should live for approximately 42 time steps. In our simulations, typically 20 to 30 reached this age or older. This is clearly an effect of selection, but it is surprisingly small. The latter is due to the fact that during a simulation we approach punctuated equilibrium for rather long intervals. This causes the fact that all animals are nearly equal, so they are again chosen at random. Among short living animals we see a better hint at the positive effect of selection. 40,000 animals should die right after their creation if they were randomly chosen, but in our simulations this number was larger than 50,000 and sometimes even reached 60,000, which cannot be attributed to statistical fluctuations. So we see that most mutations produce less fit animals just like in real nature.

Acknowledgments

We would like to thank Peter Grassberger for proposing the subject, for many stimulating and helpful discussions, and also for a critical reading of this manuscript. One of us (H. F.) wants to thank the organizers of the Institute for Scientific Interchange research workshop on "Complexity and Evolution" for their hospitality. Part of this work was conducted while attending the workshop, held in Villa Gualino, Torino, from April 17 to July 31, 1990. A lot of discussions with other participants—in particular with Steen Rasmussen, Walter Fontana, and Norman H. Packard—gave deeper insight into the problems and knowledge about related works. We especially want to thank Aviv Bergman, who critically read this paper and helped with his knowledge about population genetics. Last but not least we would like to thank the referee for giving us helpful hints concerning the main structure of this work.

References

[1] M. Eigen and P. Schuster, *The Hypercycle* (Berlin, Springer Verlag, 1979).

[2] See S. A. Kauffman, "Autocatalytic Sets of Proteins," *Journal of Theoretical Biology*, **119** (1986) 1–24; J. D. Farmer, S. A. Kauffman, and N. H. Packard, "Autocatalytic Replication of Polymers," *Physica*, **D22** (1986) 50–67; R. J. Bagley, J. D. Farmer, S. A. Kauffman, N. H. Packard, A. S. Perelson, and I. M. Stadnyk, "Modeling Adaptive Biological Systems," *BioSystems*, **23** (1989) 113–138.

[3] S. Rasmussen, C. Knudsen, R. Feldberg, and M. Hindsholm, "The Coreworld: Emergence and Evolution of Cooperative Structures in a Computational Chemistry," *Physica*, **D42** (1990) 111–134.

[4] Walter Fontana, "Algorithmic Chemistry: A Model for Functional Self-Organization," Technical Report LA-UR-90-1959, Los Alamos National Laboratory (1990).

[5] D. E. Goldberg, B. Korb, and K. Deb, "Messy Genetic Algorithms: Motivation, Analysis, and First Results," *Complex Systems*, **3** (1989) 493–539.

[6] For example, see page 65 in D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, MA, Addison-Wesley, 1989).

[7] Stephen T. Barnard and Aviv Bergman, "Adaption in Signal Space," SRI International preprint submitted to the International Workshop on Parallel Problem Solving from Nature (1990).

[8] Aviv Bergman and Marcus W. Feldman, "More on Selection for and against Recombination," *Journal of Theoretical Population Biology* (forthcoming).

[9] See John H. Holland, *Adaptation in natural and artificial systems* (Ann Arbor, University of Michigan Press, 1975); David E. Goldberg, "Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction," *Complex Systems*, **3** (1989) 129–152.