Biol. Cybern. 69, 269–274 (1993)

Biological Cybernetics

Self-replicating sequences of binary numbers. Foundations I: General

Wolfgang Banzhaf *

Central Research Laboratory, Mitsubishi Electric Corporation, 1-1, Tsukaguchi Honmachi 8-chome, Amagasaki 661, Japan

Received: 19 May 1992/Accepted in revised form: 19 March 1993

Abstract. We propose the general framework of a new algorithm, derived from the interactions of chains of RNA, which is capable of self-organization. It considers sequences of binary numbers (strings) and their interaction with each other. Analogous to RNA systems, a folding of sequences is introduced to generate alternative two-dimensional forms of the binary sequences. The two-dimensional forms of strings can naturally interact with one-dimensional forms and generate new sequences. These new sequences compete with the original strings due to selection pressure. Populations of initially random strings develop in a stochastic reaction system, following the reaction channels between string types. In particular, replicating and self-replicating string types can be observed in such systems.

1 Introduction

In this contribution we want to study an algorithm derived from prebiotic evolution. Its investigation was suggested by the application of genetic algorithms (GAs) (Rechenberg 1973; Holland 1975; Schwefel 1981; Goldberg 1989) to problems of combinatorial optimization.

In a particular formulation of the GA based on notions from molecular biology (Dewdney 1985; Wang 1987; Banzhaf 1990a) a solution to the optimization problem is represented as a string of data. Improvements of a solution during the optimization process are then achieved by machines operating on the data string. An entire population of solutions with different qualities is processed at the same time in these algorithms, giving rise to explicit parallelism in the space of data strings and machines.

The right choice of a representation for the optimization problem to be solved is crucial for the success or failure of the GA. Strings have to fulfil certain feasibility criteria for a general solution to the problem. To select a promising representation requires detailed knowledge about the particular optimization problem. The set of data strings can be initialized with random numbers.

The second ingredient of a GA is the availability of various types of machines that constantly process data strings. Processing one (or more) string(s) means that a machine picks up a solution(s) to the problem and tries to improve it (them) with the help of its operation program. Here, again, knowledge about the particular optimization problem is required, namely, in the form of programs for machines which try to improve solutions. A considerable part of these programs, however, can also be left to random number generators.

The GA proceeds, therefore, by stochastically searching in parallel for better solutions to the optimization problem than those already given in the form of initially generated data strings. If the quality of a string newly generated by a machine is better than its "parent" or predecessor from which it was derived, it is released into the ensemble of strings; otherwise it is discarded. Note that this selection process is completely local in the sense that it depends only on the string and its variant (Banzhaf 1990a, b). Nevertheless, during the processing of strings, one can observe a nearly continuous improvement of solutions until the algorithm finally settles into a local or global optimum of the problem.

If we take a look at real evolutionary processes, we can observe the following differences:

1. A living organism has to adapt to a dynamically changing environment, whereas an artificial optimization process usually adapts to a fixed selection criterion determined by the optimization problem.
2. Organisms live in an open environment and therefore have to invent and sustain metabolism. Data strings in an optimization algorithm, on the other hand, are isolated from the underlying material substrate and usually exist in a closed world.
3. Conserving and reproducing the accumulated knowledge or information is a must for living beings, and thus a value in itself. In fact, one may say that the mechanisms developed for this purpose constitute the very essence of life. In contrast, the degree of reproduction for the solutions to an optimization problem is in the average case

Dedicated to Professor Hermann Haken on the occasion of this 65th birthday

* *Present address:* Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA

completely controlled by the performance of a string with respect to a single criterion, the particular optimization problem to be solved.

In living organisms, the main goals (sustained metabolism and reproduction) are achieved by (1) information storage (in double strands of DNA), which could be said to carry solutions to the problems of life (the genotype) and (2) active functional units (proteins etc.) for performing various operations in the real world (the phenotype), e.g., reproduction or metabolism.

The entire system providing all these abilities must be very complex, since the information about how to conserve information has also to be conserved in the DNA itself (e.g., Lewin 1987). Naturally, this provokes the question: what came first, DNA or proteins? What was long the subject of conjecture was finally confirmed a decade ago (Kruger et al. 1982; Guerrier-Takada et al. 1983): there exists a simpler system, based on RNA, capable of both information storage *and* enzyme-like operations. This system is visible in higher organisms even today, when it has been fully integrated by performing many useful auxiliary functions for the DNA-protein system. The most common plausible sequence of events leading to the present-day replication-transcription-translation machinery is: unknown precursor template molecule $\Rightarrow$ RNA $\Rightarrow$ RNA + protein $\Rightarrow$ RNA + protein + DNA.

As proteins RNA has a natural tendency to fold itself into a secondary and tertiary structure, depending on the sequence of nucleotides (amino acids in the case of proteins) in the strand. The only difference between RNA and DNA is that the ribose moiety causes it to prefer the "A" conformation of the double helix. Structure formation in RNA is driven by the formation of intramolecular RNA double helices. Folded RNA molecules have been shown to be protein-like, occasionally able to perform operations either on themselves ("ribozyme") or on other strands of RNA. Hence, the sequence on a strand of RNA can be interpreted as information, whereas the three-dimensional shape of the RNA molecule resulting from the attractive and repulsive forces of physics determines its operational counterpart, that is, its function.

If we now turn to the world of modern computers, we can state that sequences of binary numbers, which we shall here call binary strings, are the simplest form of information storage and representation. What we shall do in this study is to consider a population of binary strings in which strings can act, much like RNA, as *both* information storage (strings) *and* machines (operators) changing strings. This will require the introduction of at least a second form of strings, analogous to the secondary or tertiary form that primary amino acid (or nucleotide) sequences can assume. The resulting algorithm cannot be considered a GA anymore. Rather, it is a prebiotic algorithm, concerned with the onset of replication in an ensemble of artificial macromolecules (the strings of numbers).

On a higher level of organization, Fontana has recently proposed a model called *algorithmic chemistry* (Fontana 1991), in which he considers similar interactions between strings. The strings in his model, however, are taken from pure LISP (Chaitin 1987), a special version of the LISP computer language inspired by Church's λ-calculus. Interestingly, similar phenomena of self-organization occur in both models. Other important work in this context has been done by Kauffman, who proposed simplified models of autocatalytic sets of proteins (Kauffman 1986).

This paper is the first part of a series organized as follows. Part I describes the general algorithm for an interaction between the two forms of strings and introduces the basic notions necessary to understand the dynamics of such a system. Part II (Banzhaf 1993a) discusses the simplest nontrivial system of this kind with string length $N = 4$. Static and dynamic features of this system are studied using selected simulations. It also outlines our efforts to model the simple system by deterministic rate equations. Finally, Part III (Banzhaf 1993b) considers more complicated systems with larger strings. Simulations are presented on the $N = 9$ system. Also in this part, mutations of strings are introduced and the existence of evolutionary processes is demonstrated.

2 The basic algorithm

As we deal with the evolution of strings of numbers or symbols, two principles have to be embodied in the algorithm:

1. Machines, which we shall call *operators*, should exist, able to change strings according to specified rules.
2. Strings should be equivalent to machines in that a predefined mapping law determines which operator can be generated from which string type.

Since we want to construct an algorithm as simple as possible, we restrict ourselves here to strings of binary numbers. Therefore, let us consider string $\mathbf{s}$, consisting of concatenated numbers 0 and 1:

$$s = (s_1, s_2, \ldots, s_i, \ldots, s_N), \; s_i \in \{0, 1\}, \; 1 \leqslant i \leqslant N \,. \qquad (1)$$

For N, the length of strings, we adopt a square number, at least in this contribution.

The restriction for s_i to be a binary number is by no means necessary, but it will serve us in the course of our discussion to concentrate on some of the essential points. In principle, real numbers and even strings of symbols may be considered, provided appropriate (and, maybe, natural) rules can be found that govern the formation of operators and their interaction with strings.

This brings us to the decisive question: how can these operators be formed from binary strings? In nature, amino acid strands tend to fold together according to the laws of chemistry and physics. Although it is quite clear that formation is governed mainly by a tendency of the strands to relax into energy-minimal configurations in three dimensional physical space, studying the formation of secondary and tertiary structures which result from primary amino acid sequences is in itself a field of vigorous scientific interest (Robson and Garnier 1986; Bryngelson and Wolynes 1987; Li and Scheraga 1987; Quian and Sejnowski 1988 and references therein).

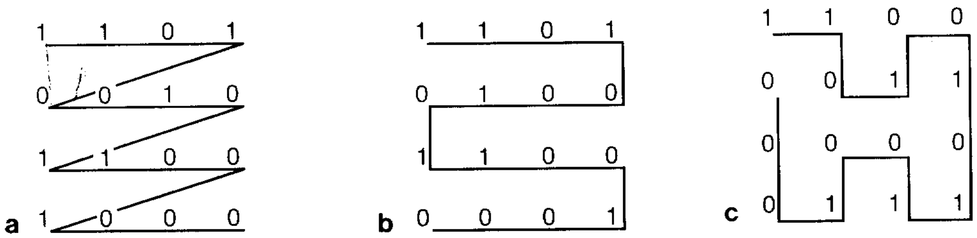

Fig. 1a–c. Some two-dimensional compactly folded forms of a string in an example with $N = 16$ binary numbers: $s = (1101001011001000)$. **a** Nontopological folding; **b, c** topological foldings

2.1 Folding of sequences

For binary strings we propose the following procedure. A string s with N components folds into an operator $\mathcal{P}$ which can be represented mathematically by a quadratic matrix of size $\sqrt{N} \times \sqrt{N}$, as is schematically depicted in Fig. 1 for three particular foldings. At this stage of the discussion, it is not necessary to fix the exact method of string folding. Any kind of mapping

$$\mathcal{M}: s \rightarrow \mathcal{P}_s \tag{2}$$

of the topologically one-dimensional strings of binary numbers into a two-dimensional (quadratic) array of numbers is allowed. Depending on the method of folding, we can expect various transformation paths between strings.

Computationally, the proposed folding methods are equivalent to the treatment of polar and nonpolar residue sequences studied by Lau and Dill (1989). They consider the conformation of globular proteins as the outcome of a self-avoiding random walk on a two-dimensional square lattice. The compact conformations they examine as candidates for minimal energy configurations precisely correspond to the square matrices in our model.

Let us assume now that an operator $\mathcal{P}_s$ was formed from string s. This operator, in turn, can act on another string s' and generate still another string s'' (see Fig. 2):

$$\mathcal{P}_s s' \Rightarrow s''. \tag{3}$$

Specifically, we require that neither $\mathcal{P}_s$ nor s' is *deleted* by this operation.[1] Rather, a new string s'' is generated by the cooperation of $\mathcal{P}_s$ and s'. In other words, the system is open with an ongoing production of new strings from some sort of raw material (corresponding to energy-rich monomers in nature). In this interpretation, only the *information* carried by $\mathcal{P}_s$ and s' is considered important here, and this information is required to be conserved. The options to control the continued production of new strings are the following:

1. One can run the system with a fixed number of strings M. Each new string produced causes the deletion of an already existing one. The string to be replaced can be selected by chance, i.e., according to its frequency in the ensemble; or by some quality criterion, length of string, number of 1s, etc.; or by a combination of both.
2. One can fix a maximum number of strings M_2 after allowing for an initial period of unrestricted growth started from a small number of strings M_1.
3. One can restrict the raw material that may be used to produce strings. As a consequence, an initial growth period in the number of strings causes a rapid depletion in the supply of raw materials, in our case 0s and 1s, which, in turn, restricts the formation of new strings.

The net effect of these or other countermeasures is to force strings into a fierce competition for available resources (Eigen 1971). Strings with selective advantages will be preferred strongly by the system.

2.2 Operators at work

For the moment, however, we have to come back to the question of exactly how an operator can act on a string. Consider Fig. 2. We can think of s' as being concatenated from $\sqrt{N}$ segments with length $\sqrt{N}$ each. The operator $\mathcal{P}_s$ is able to transform one of these segments at a time, using semilocal operations. In this way, it moves down the string in steps of size $\sqrt{N}$ until it has finally completed the production of a new string s''. Then, operator

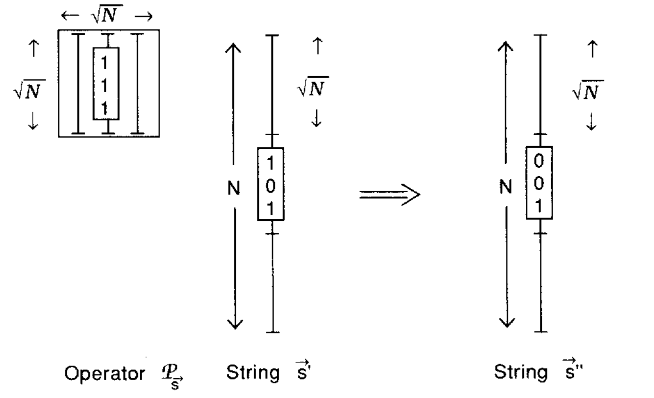

Fig. 2. An operator $\mathcal{P}_s$ of matrix dimension $\sqrt{N} \times \sqrt{N}$ (derived from string s) acts on a string s' consisting of $\sqrt{N}$ segments, each of length $\sqrt{N}$, to produce a new string s''

[1] This is the symmetric case we consider in further detail here. It is also possible to require that only one of either string or operator should be conserved. Qualitatively, the behavior of such a system is similar

$\mathscr{P}_s$ unfolds back into its corresponding form as a string s and is released, together with s' and s'', into the ensemble of strings. (We will call this "*string soup*.") Thus, we have to require that the mapping from one-dimensional strings into two-dimensional operators is *reversible*.

The semilocal character of operations is an important feature of this model as it enables the interaction of similar operator and string pairs to result in similar product strings. We call the operations semilocal, since their interaction range is restricted to the size of one dimension of the operator form of a string $\sqrt{N}$. Interestingly, it is the folding mechanism itself which causes this restriction in the interaction range.

A particular example of the action of an operator onto a string is the computation of scalar products. This computation, however, will not conserve the binary character of strings unless we introduce a nonlinearity. Therefore, in Part II (Banzhaf 1993a) we shall examine in more detail the following related computation:

$$s'_{i+k\sqrt{N}} = \sigma\left[\sum_{j=1}^{j=\sqrt{N}} \mathscr{P}_{ij} s_{j+k\sqrt{N}} - \Theta\right]$$

$$i = 1, \ldots, \sqrt{N} \quad k = 0, \ldots, \sqrt{N} - 1 \,. \quad (4)$$

The symbol $\sigma[\ ]$ stands for the squashing function

$$\sigma[x] = \begin{cases} 1 & \text{for } x \geqslant 0 \\ 0 & \text{for } x < 0 \end{cases} \quad (5)$$

and Θ is an adjustable threshold. For the case $\Theta = 1$ discussed in Parts II and III (Banzhaf 1993a, b), (4) amounts to a combination of boolean operations, applied separately in each segment k of the string.

In order to gain a better understanding of the entire system, it helps to adopt the reaction notation of chemistry. In this picture, consider an operator $\mathscr{P}$, formed from $s^{(1)}$, which reacts with $s^{(2)}$ to produce $s^{(3)}$ under conservation of all reactants.[2] We can write this as a general reaction of the kind

$$s^{(1)} + s^{(2)} + X \rightarrow s^{(1)} + s^{(2)} + s^{(3)} \quad (6)$$

and classify the possible reactions into five classes:

$$\mathscr{P}_{s^{(1)}} \oplus s^{(2)} \rightarrow s^{(1)} + s^{(2)} + s^{(3)} \quad (7)$$

$$\mathscr{P}_{s^{(1)}} \oplus s^{(2)} \rightarrow s^{(1)} + s^{(2)} + s^{(2)} \quad (8)$$

$$\mathscr{P}_{s^{(1)}} \oplus s^{(2)} \rightarrow s^{(1)} + s^{(2)} + s^{(1)} \quad (9)$$

$$\mathscr{P}_{s^{(1)}} \oplus s^{(1)} \rightarrow s^{(1)} + s^{(1)} + s^{(2)} \quad (10)$$

$$\mathscr{P}_{s^{(1)}} \oplus s^{(1)} \rightarrow s^{(1)} + s^{(1)} + s^{(1)} \quad (11)$$

where operators $P_{s^{(k)}}$ are characterized by the strings to which they correspond. The symbol $\oplus$ indicates the active process of the operator working on a string (using X as raw material). Interesting reactions are given in (8), (9), and (11), which show replication of one reactant (the former two) and self-replication (the latter).

2.3 Equilibration

We have to mention the fact that potentially "lethal" strings exist in these systems. A string is said to be lethal or "pathological" if it is able to replicate in an unproportionally large number in almost any ensemble configuration. In the particular case of (4), the string consisting of only 0s is pathological because it is able to replicate with itself and with every other string. We call this string the *destructor* and shall constantly monitor string soup reactions in order to remove the destructor on appearance. Another potentially hazardous string consists of 1s only. We call it the *exploitor*. In addition to replicating itself it is able to replicate with a large fraction of string types. Although the exploitor is pathological, we can deal with it in a more gentle way by providing a means of nondeterministic string removal.

To this end, for example, we introduce the following general stability criterion for strings: the fewer number of 1s a string contains, the more stable it becomes. Its chance to be removed, therefore, depends on

$$I^{(k)} = \sum_{i=1}^{N} s_i^{(k)}, \quad k = 1, \ldots, M \quad (12)$$

$I^{(k)}$ measures the number of 1s in string k and will determine a probability

$$p^{(k)} = [I^{(k)}/N]^n \quad (13)$$

with which an encountered string should be removed. The parameter n shall serve us to adjust probabilities slightly. Note that the exploiter has probability $p = 1$ and must disappear on encounter.

2.4 Summary

The simplest version of the entire algorithm can now be stated as follows:

Step 1: Generate M_1 random binary strings of length N each.
Step 2: Select a string and fold it into an operator (a matrix) of dimension $\sqrt{N} \times \sqrt{N}$.
Step 3: Select another string and apply the operator generated in step 2.
Step 4: Release the new string, the old string, and the operator (as string) into the string soup.
Step 5: Remove one randomly chosen string with probability $p = \dfrac{M_1}{M_2}$ in order to compensate for the addition of a string in step 4.
Step 6: Monitor the soup and replace destructors with randomly generated strings.[3]

[2] In the future, when discussing string types, we shall omit the vector arrow and only use numbers to characterize different string types

[3] Randomly generated strings could either be completely random binary strings or result from a mutation of already existing strings

Table 1. Some low-dimensional examples

$\sqrt{N}$	N	n_S	n_R
2	4	15	210
3	9	511	$\sim 2.6{\cdot}10^5$
4	16	65535	$\sim 4{\cdot}10^9$
5	25	$\sim 10^7$	$\sim 10^{15}$
10	100	$\sim 10^{30}$	$\sim 10^{60}$

$\sqrt{N}$, Matrix size in one dimension; N, length of strings; n_S, number of different strings, excluding destructor; n_R, number of possible reactions, excluding self-reactions

Step 7: Select one string and substitute it according to the probability of (13), by a randomly generated string.[3]

Step 8: Go to step 2.

This algorithm implements the resource limitation scenarios previously mentioned as control option 1, $M_1 = M_2 \equiv M$, or 2, $M_1 < M_2$ (see Sect. 2.1).

Table 1 shows the impressive number of possible interactions between strings as we increase their length N. For arbitrary N we have

$$n_S = 2^N - 1 \tag{14}$$

string types and

$$n_R = 2^{2N} - 3{\cdot}2^N + 2 \tag{15}$$

different reactions, excluding reactions with the destructor and self-reactions. The number of potential self-replications n_{SR} is, of course,

$$n_{SR} = n_S \tag{16}$$

3 Conclusion

In this contribution we have discussed a new algorithm for a self-organizing system based on sequences of binary numbers. The key element of this algorithm is the proposed mechanism of interaction between strings. It is based on a two-dimensional form that the strings can assume by application of various folding methods. In contrast to other models (Fontana and Schuster 1987; Fontana et al. 1989) the folding method as well as the operations of the two-dimensional (matrix) form of strings are *not* based on physical and chemical effects. The logic of our model seems to be much simpler than the physical logic of RNA folding in nature.

There is, of course, a lot of room to make the proposed framework more elaborate, for instance, by including local attractive forces between string components. Another possibility would be to provide an additional folding step of two-dimensional forms into three-dimensional "artificial globular molecules".

Any introduction, however, of new elements into the rules of string folding must not destroy the operating principle that new strings are produced by an interaction of operators and strings. Whether we require that both interacting entities or only one be conserved seems unimportant relative to the general interpretation of the process as being an open nonequilibrium process (Haken 1983). This is, after all, the implication of our requirement that strings can be assembled utilizing the information generated by the operator-string interaction.

The interactions between string types generate a reaction network, and we expect broad differences in the production rates of various types of strings. In particular, replicating or self-replicating types may feature prominently. In a population of strings this will lead to a self-organizing dynamic with changing relative frequencies of string types.

Our next step will be a computer study of the proposed algorithm using a population of strings of shortest length, $N = 4$. This will constitute the second part of our series (Banzhaf 1993a).

Acknowledgements. I wish to thank my colleague Mr. T. Iwamoto, who reignited my interest in autocatalytic systems. I am very grateful to Drs. T. Nakayama and K. Kyuma for establishing and maintaining the creative atmosphere in our research group without which this work never would have been possible. I would also like to thank the members of our reserach unit for valuable discussions. I am indebted to Mr. S. Winquist for carefully reading an earlier version of this manuscript.

References

Banzhaf W (1990a) The "molecular" travelling salesman. Biol Cybern 64:7–14

Banzhaf W (1990b) Finding the global minimum of a low-dimensional spin-glass model. In: Voigt HM, Mühlenbein H, Schwefel HP (eds) Selected papers on evolution theory, combinatorial optimization and related topics, Akademie, Berlin

Banzhaf W (1993a) Self-replicating sequences of binary numbers. Foundations II: Strings of length $N = 4$. Biol Cybern 69:275–281

Banzhaf W (1993b) Self-replicating sequences of binary numbers. Foundations III: Larger systems. Biol Cybern (submitted)

Bryngelson J, Wolynes P (1987) Spin glasses and the statistical mechanics of protein folding. Proc Natl Acad Sci USA 84:7524–7528

Chaitin GJ (1987) Algorithmic information theory. Cambridge University Press, Cambridge, UK

Dewdney AK (1985) Computer recreations. Sci Am 257(3): 21–32

Eigen M (1971) Self-organisation of matter and the evolution of biological molecules. Naturwissenschaften 58:465–523

Fontana W (1991) Algorithmic chemistry. In: Langton CG, Taylor C, Farmer JD, Rasmussen S (eds) Artificial life 2. Addison-Wesley, Redwood City, Calif

Fontana W, Schuster P (1987) A computer model of evolutionary optimization. Biophys Chem 26:123–147

Fontana W, Schnabl W, Schuster P (1989) Physical aspects of evolutionary optimization and adaptation. Phys Rev A40:3301–3321

Goldberg D (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading, Mass

Guerrier-Takada C, Gardiner K, Marsh T, Pace N, Altman S (1983) The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme. Cell 35:849–857

Haken H (1983) Synergetics. An Introduction. Springer, Berlin

Holland JH (1975) Adaption in natural and artificial systems. University of Michigan Press, Ann Arbor

Kauffman S (1986) Autocatalytic sets of proteins. J Theor Biol 119:1–24

Kruger K, Grabowski PJ, Zaug AJ, Sands J, Gottschling DE, Cech TR (1982) Self-splicing RNA: autoexcision and autocyclization of the ribosomal RNA intervening sequence of *Tetrahymena*. Cell 31:147–157

Lau KF, Dill KA (1989) A lattice statistical mechanics model of the conformational and sequence spaces of proteins. Macromolecules 22:3986–3997

Lewin B (1987) Genes, 3rd edn. Wiley, New York

Li Z, Scheraga H (1987) Monte Carlo-minimization approach to the multiple-minima problem in protein folding. Proc Natl Acad Sci USA 84:6611–6615

Quian N, Sejnowski T (1988) Predicting the secondary structure of globular proteins using neural network models. J Mol Biol 202:865–884

Rechenberg I (1973) Evolutionsstrategien. Frommann-Holzboog, Stuttgart

Robson B, Garnier J (1986) Introduction to proteins and protein engineering. Elsevier, Amsterdam

Schwefel HP (1981) Numerical optimization of computer models. Wiley, Chichester

Wang Q (1987) Optimization by simulating molecular evolution. Biol Cybern 57:95–101