Brill Academic Publishers
P.O. Box 9000, 2300 PA Leiden,
The Netherlands

Lecture Series on Computer and Computational Sciences
Volume 1, 2005, pp. 1-6

Genetic Algorithm Evolution of Cellular Automata Rules for Complex Binary Sequence Prediction

A. V. Adamopoulos[(a),1], N.G. Pavlidis[(b)] and M.N. Vrahatis[(b)]

[(a)] **Medical Physics Laboratory, Department of Medicine, Democritus University of Thrace, GR-681 00 Alexandroupolis, Hellas**

[(b)] **Department of Mathematics and Computational Intelligence Laboratory, University of Patras, GR-26110 Patras, Hellas.**

Abstract: Complex binary sequences were generated by applying a simple threshold, linear transformation to the *logistic iterative function*, $x_{n+1} = r\,x_n\,(1-x_n)$. Depending primarily on the value of the *non-linearity parameter* r, the logistic function exhibits a great variety of behavior, including stable states, cycling and periodical activity and the period doubling phenomenon that leads to high-order chaos. Binary sequences of length $2L$ were used in our computer experiments. The first L bits (first half) were given as input to Cellular Automata (CA) with the task to regenerate the remaining L bits (second half) of the binary sequence in less than L evolution steps of the CA. To perform this task a suitably designed Genetic Algorithm (GA) was developed for the evolution of CA rules. Various complex binary sequences were examined, for a variety of initial values of x_0 and a wide range of the non-linearity parameter, r. The proposed hybrid prediction algorithm, based on a combination of GAs and CA proved quite efficient.

Keywords: Cellular Automata, Genetic Algorithms, Complex Binary Sequence Prediction

Mathematics Subject Classification:

1 Introduction

Cellular Automata (CA) are decentralized structures of simple and locally interacting elements (cells). A CA starts with a given initial configuration that refers to the initial state of its cells. A CA evolves following a set of rules that incorporate [1, 2]. This set of rules yields the derived cell states (values) at the next evolution step, for all the possible combinations of cell states. CA have been proposed as a novel approach for a large number of problems. Among others, CA have been proposed as models for physical, biological and social systems, games and pattern recognition [3, 4]. As systems, CA have proved capable of parallel and emergent computation [5, 6]. In this work, simple, bistable, one-dimensional CA were used for the purpose of prediction of binary sequences of high complexity. The considered binary sequences were derived by applying a linear threshold transformation, proposed in [7], on data obtained from the logistic function. The task under consideration for the CA was to use the first half of a given binary sequence as input (initial configuration), in order to reproduce the second half of that sequence in a given number of evolution steps of the CA. For that purpose, a population of CA was evolved using a suitably designed Genetic Algorithm (GA) [8, 9]. Namely, the GA evolved the set of rules of the

[1] Corresponding author. Email adam@med.duth.gr

CA [10–12]. Results show that the proposed method can provide CA rules that give in a small number of evolution steps a 100% prediction of long binary sequences.

2 Materials and Methods

The main task of this work is to utilize a GA to evolve CA rules, so that the CA is capable of predicting a complex binary sequence in a certain number of evolution steps. Complex binary sequences were generated using a two step procedure. At the first step, the logistic function:

$$x_{n+1} = r\, x_n\, (1 - x_n), \tag{1}$$

was used, for the generation of sequences of real numbers of length $2L$. The choice of the logistic function to obtain data was not arbitrary; on the contrary, it was based on the highly complex behavior it exhibits. Specifically, for values of the non-linearity parameter r in the range $[0, 3)$ the system reaches a single-state stable value. For r in the range $[3, 3.57)$ the period doubling phenomenon occurs, and the system exhibits cycling (periodical) behavior with increasing cycling period as the value of r increases. This results to a fully chaotic behavior for even larger values of r in the range $[3.57, 4]$.

After the generation of the chaotic data, x, using Eq. (1), at a second step, the binary sequence $b_n(x_0, r)$ is generated by applying the transformation [7]:

$$b(x_0, r) = \begin{cases} 0, & \text{if } x_n \leqslant 0.5 \\ 1, & \text{if } x_n > 0.5 \end{cases} \tag{2}$$

Taking a binary sequence $b_n(x_0, r)$ of length $2L$ bits, the first half (first L bits) were used as input to the CA. In other words, these L bits were used for the construction of the initial configuration of the CA, which corresponds to the evolution step zero. The task was to let the CA evolve using the set of rules for a number of evolution steps less than or equal to L and to investigate if the CA was able to regenerate the second half of $b_n(x_0, r)$, namely the last L bits of $b_n(x_0, r)$. The number of rules of a specific CA depends on the order R (radius) of cell neighborhood. This parameter determines the number $2R+1$ of cells that a specific cell interacts with in a local manner. Thus, for $R = 1$, the neighborhood of each cell consists of three cells. This is shown in Table 1. A three-cell neighborhood, with each cell considered as a bistable element, may present $C = 2^{2R+1} = 2^3 = 8$ distinct combinations. Thus, the adaptation of a set of 8 rules is necessary. In the general case, these C combinations are ordered and numbered from 0 to $C - 1$ following the representation of integer numbers in the binary arithmetical system.

Rule Nr. C:	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
	1	0	0	1	1	0	1	0

Table 1: An example of a set of rules for $R = 1$, which results to $C = 8$ distinct rules.

To accomplish our task, the sets of rules of a population of N CA were evolved using a GA. Each set of rules was represented as the chromosome of the individuals of the GA, with each gene being a binary digit. The length of the chromosome was C, as this is the number of rules that comprise the corresponding set of rules. The binary representation of the GA individuals allowed the use of well known and widely used genetic operators for selection, crossover and mutation [8]. In particular, as selection operator we used the *roulette-wheel selection operator*; as crossover operator we used the *one-point crossover operator*; and finally, as mutation operator we used the *bit-flip operator*.

Recalling our goal, as fitness function for the evaluation of the individuals of the GA we used the number of bits that were successfully predicted after L evolution steps of the CA.

The necessity of utilizing a GA must be emphasized. As it is shown in Table 2, the size V of the search space is enormously expanding, even for small values of R.

Radius R	Number of neighbors H	Number of rules C	Size of search space V
1	3	8	$2^8 = 256$
2	5	32	$2^{32} = 4294967296$
3	7	128	$2^{128} = 3.40282 \cdot 10^{38}$
4	9	512	$2^{512} = 1.34078 \cdot 10^{154}$

Table 2: Number of possible combinations of CA rules for various values of R.

3 Results

The proposed method was tested for various binary sequences which were generated for a large number of combinations of the parameters x_0 and r of the logistic equation in Eq. (1). The values of the non-linearity parameter, r, were selected in the range $[3.57, 4]$ for which chaotic behavior is exhibited by Eq. (1).

The values of half-length L of the binary sequence reached up to 50. That is, a sequence of 50 bits was given as input (initial configuration) to a CA, in order to regenerate the next 50 bits of the given binary sequence. Typically, the GA employed 50 to 200 individual, the crossover probability was in the range $[0.9, 1.0]$, the mutation probability was in the range $[0.05, 0.2]$, and the GA evolved for 20 up to 500 consecutive generations.

The evolution for 40 GA generations of the fitness function of the individual with the highest fitness function value at each generation for an experiment with $L = 20$ is shown in Fig. 1. As it is shown, the GA was able to track very quickly (in less than 40 GA generations) a suitable set of rules that regenerate the desired output in less than 20 evolutionary steps of the CA.

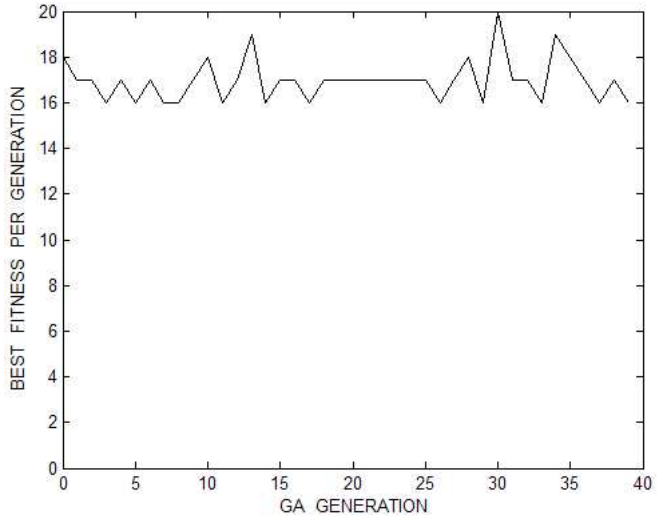

Figure 1: Evolution of the best fitness value of each GA generation, for a binary sequence with $L = 20$.

4 Discussion

In the present work a hybrid evolutionary algorithm was proposed for the prediction of compex binary sequences. The algorithm incorporates Cellular Automata with sets of rules that are suitably codified in order to be evolved by a Genetic Algorithm. Our purpose was to test the method in binary sequences of high complexity, like the ones that are obtained by applying a simple, linear threshold transformation on the iterative logistic function of Eq. (1), which is a well-known system that through period doubling reaches to chaotic behavior. The proposed algorithm was given an L number of bits as input and its task was to evolve the sets of rules of the CA in order to regenerate the next L bits of the binary sequence in less than L evolution steps of the CA. The obtained results for values of L up to 50 and for a variety of values of the non-linearity parameter r of Eq. (1) indicated that the proposed algorithm was able to find the proper sets of CA rules in a small number of GA generations.

Acknowledgment

This work was partially supported by the Hellenic Ministry of Education and the European Union under research Program PYTHAGORAS-89203.

References

[1] S. Wolfram: *Theory and Applications of Cellular Automata*, World Scientific, 1986.

[2] S. Wolfram: *Nature*, Vol. 311, 1984, pp. 419–424.

[3] S. Wolfram: *Cellular Automata and Complexity*, World Scientific, Singapore, 1994.

[4] N. Ganguly, B.K. Sikdar, A. Deutsch, G. Canright, P.P. Chaudhuri: A Survey on Cellular Automata, Technical report, Centre for High Performance Computing, Dresden University of Technology, December 2003.

[5] M. Mitchell. Computation in Cellular Automata. Non-standard computation, p. 95–140, 1998.

[6] J.P. Crutchfield and M. Mitchell, Proceedings of the National Academy of Sciences, USA, 92(23), 10742.

[7] N.H. Packard, Complex Systems **4**, 543 (1990).

[8] M. Mitchell: *Introduction to Genetic Algorithms*. MIT Press (1996).

[9] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[10] M. Mitchell, J. Crutchfield, and R. Das, Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work, In: First International Conference on Evolutionary Computation and its Applications, 1996.

[11] M. Mitchell, J.P. Crutchfield, and P.T. Hraber, Physica D 75, 361–391, 1994.

[12] R. Das, J.P. Crutchfield, M. Mitchell and J.E. Hanson, Evolving globally synchronized cellular automata, In: L.J. Eshelman (ed.), Proceedings of the Sixth International Conference on Genetic Algorithms 336–343. San Francisco, CA Morgan Kaufmann, 1995.

Appendix

A one-dimensional cellular automaton consists of a lattice of N identical finite-state machines (cells), each with an identical topology of local connections to other cells for input and output, along with boundary conditions. Let Σ denote the set of states in a cell's finite-state machine and let $k = |\Sigma|$ denote the number of states per cell. Each cell is indexed by its site number $i = 0, 1, \ldots, N-1$.

A cell's state at time t is denoted by s_i^t, where $s_i^t \in \Sigma$. The state s_i^t of cell i together with the states of the cells to which it is connected is called the *neighborhood*, η_i^t, of cell i. Each cell obeys the same transition rule $\phi(\eta_i^t)_i)$, that gives the update state $s_i^{t+1} = \phi(\eta_i^t)$ for cell i as a function of η_i^t. We will drop the indices on s_i^t and η_i^t when we refer to them as general (local) variables.

We use s^t to denote the configuration of cell states: $s^t = \{s_0^t, s_1^t, \ldots, s_{N-1}^t\}$. A CA $\{\Sigma^N, \phi\}$ thus specifies a global map Φ of the configurations:

$$\Phi : \Sigma^N \to \Sigma^N,$$

with

$$s^{t+1} = \Phi(s^t).$$

In some cases in the discussion below, Φ will also be used to denote a map on subconfigurations of the lattice. Whether Φ applies to global configurations or subconfigurations should be clear from context. In a synchronous CA, a global clock provides an update signal for all cells: at each t all cells synchronously read the states of the cells in their neighborhood and then update their own states according to $s_i^{t+1} = \phi(\eta_i^t)$.

The neighborhood η is often taken to be spatially symmetric. For one-dimensional CAs, $\eta_i = s_{i-R}, \ldots, s_0, \ldots, s_{i+R}$, where R is the CA's radius. Thus, $\phi : \Sigma^{2R+1} \to \Sigma$. For small-radius, binary-state CAs, in which the number of possible neighborhoods is not too large ϕ is often displayed as a look-up table, or rule table, that lists every possible η together with the *output bit* s_i^{t+1}.

One-dimensional CAs, with $(k, r) = (2, 1)$ were used in the presented work. Here, the neighborhood of each cell consists of itself and its two nearest neighbors and the boundary conditions are periodic: $s_N = s_0$. An example is shown in the following figure.

Rule table ϕ:

Neighborhood η:	000	001	010	011	100	101	110	111
Output bit $\phi(\eta)$:	0	1	1	1	0	1	1	0

Lattice configuration:

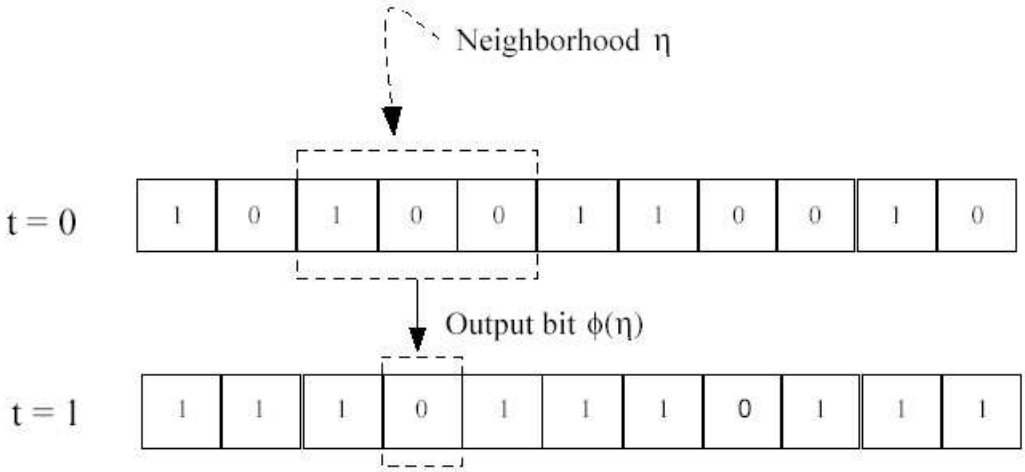

Figure 2: The components of one-dimensional, binary-state, $r = 1$ ("elementary") CA 110 shown iterated one time step on a configuration with $N = 11$ lattice sites and periodic boundary conditions (i.e. $s_N = s_0$).