Searching for the lost noise: the genetical experiments

Ioan Hălălae
"EFTIMIE MURGU" University, Faculty of Engineering, Traian Vuia Square 1-4, 320085 Resita, Romania
Phone: +40 255 210227, Fax: +40 255 210230, e-mail: i.halalae@uem.ro

Abstract. *The project presented here aims at exploring the possibility of using the CA as random number generators. We have made simulation with linear CA, with 256 cells, and used genetic algorithms for obtaining better chromosomes from the older ones.*

Keywords: *cellular automata, randomness, prediction, genetic algorithms, experimental mathematics*

1. The project presentation

Our starting point was the paper *'Looking for the lost noise'* by Gh. Stefan [13]. This paper advances the idea of using the CA as random number generators (in several configurations).

The project was developed in two phases.

The first phase aimed at monitoring the way in which classical CA evolve, that is, without using genetic algorithms. A description of this phase is included in paragraph 4.

In the second phase, we have used the chromosomes obtained in the first one, in order to make genetic experiments with chromosome colonies.

Simulation was made with linear CA with 256 cells.

The present paper presents the second phase of the project.

In *'Looking for the lost noise'* Gh. Stefan put forward the hypothesis. Since we are presenting the experiment, the title of our paper is '*Searching for the lost noise*: the genetical experiments'.

2. Modeling of parallel phenomena

2.1 The new paradigm: Concept metamorphosis

Modeling of non-deterministic phenomena has generated the apparition of new paradigms; classic concepts have undergone a series of metamorphoses. The new point of view is based on the idea of *searching* the solution for a problem.

Iterative determining of solutions is replaced by **a search process** within a space of solutions. Determining of **the optimal solution** becomes searching for an **as-good-as-possible** solution. The notion of *the best* (optimal) *solution* turns into the notion of *as-good-as-possible solution*. The iterative determining process of the optimal solution (the classical form) turns into a process of searching for an *as-good-as-possible solution, within given cost-parameters*.

Cost parameters can appear either as a physical time limit (total duration allotted to the search of the solution), or as a threshold for the number of searches. Irrespective to the form they are specified, cost parameters usually represent a decisive aspect in defining the search. Practically, beyond a certain search time, obtaining *the best solution* is too costly. To the limit, a certain solution, is to be preferred to the absence of any solution, with the motivation that the respective solution isn't the best possible!

Let us mention, too, that in large-scale spaces, *defining* 'the best' solution is a problem in itself.

The new paradigm has, as its program, *finding of some solution*, and, eventually, its *improvement* (within clear cost conditions).

2.2 Genetic Colonies

Experiments with **genetic colonies belong** to this category of **solution searching algorithms**. The basic idea of genetic algorithms is to *start from a certain initial solution and to try to improve it*.

We use some techniques for generating new solutions, starting from the initial ones. We define an evaluation and classification function for the new solutions (the 'fitness' function).

The process goes on from the new solution: a solution with better performances than the initial one is used as a starting point for a new search, and so on. Finally, the limits of the search process are set by the cost parameters.

2.3 Experimental Mathematics

Experimental Mathematics is a paradigm appearing in association "with the exploratory use of a computer" [1], especially "when one attempts to analyze experimentally algorithms" [8]. In our case, we have used the computer for simulating CA. The space of the solutions is of the 2^{256} dimension.

3. Genetic algorithm

3.1 Top-level description of a genetic algorithm

The space of the solutions where the search takes place is formed of linear uniform CA with 256 cells that can have two states. We will henceforth name them chromosomes or cellular automata, according to the context.

Let us start by reviewing the top-level description of a genetic algorithm ([5]):

1. Initialize a population of chromosome.
2. Evaluate each chromosome in that population.
3. Create new chromosomes by mating current chromosomes; apply mutation and recombination as the parent chromosomes mate.
4. Delete members of the population to make room for the new chromosomes.
5. Evaluate the new chromosomes and insert them into the population.

   If time is up, stop and return the best chromosome; if not, go to 3.

3.2 Description of a genetic algorithm

When describing a genetic algorithm, we should make precisions concerning the following aspects([5], [12]):

- how do we get the *initial population*; and, implicitly, which is the *dimension of the standard population* (the number of chromosomes the population consists of)
- *how do we get offspring* from the current population (how do we get from a 'generation' to the next one); otherwise put, how do we perform the crossing of the chromosomes
- how do we perform the *performance evaluation*, or, equivalently, how do we select the chromosomes upon which the experiment is pursued
- *how* or *when the experiment is stopped*

The term *gene pool* refers to the population we are working with, that is:

- at the first step of the experiment, to the initial population
- during the experiment, to the population obtained from applying crossings to the population from the preceding step (the preceding generation / iteration)

4. Chromosomes

4.1 Randomness and prediction

We use a definition of randomness inspired by G. Chaitin([2]-[4]): an automaton is random as long as *its evolution cannot be predicted*. From the moment when its evolution can be predicted, it is no longer random. An automaton is random *until it starts cycling*. We decide to stop the automaton in the moment it starts cycling, and to consider its evolution as complete

Practically, one way or another, we must:

- have an evidence of the states of the automaton, in the order of their appearance, and
- test a new state of the automaton (the current status, the last one resulted from the calculus process) if it appears from the first time or not.

If we find it in the 'history' of the automaton, it represents the end of the first cycle of the automaton and the generation of new states stops. If not, the new status is to be archived and the iterations continue.

We keep the whole automaton 'history' in a matrix, whose successive rows memorize the automaton's states in the order of their apparition. We will name this matrix 'the evolution matrix'.
In order to synthesize the results, we have to monitor the chromosomes in two different ways:

- the first one: we monitor for each chromosome the evolution as an internal mechanism of the cellular automaton
- the second one, we archive the results of each chromosome, individually, together with its initial configuration and the performance

And we must, of course, not repeat the tests for the same chromosome.
As regards chromosomes, there are two problems:

- the generation of the initial states
- the concrete work with a CA.

This time, we didn't perform the chromosome initialization randomly: we have systematically generated distinct chromosomes.

For actually working with a cellular automaton (monitoring the evolution of the automaton), we arrived at the following diagram:

1. An initial state is generated
2. The initial state is memorized in the evolution matrix, in line 1 of the matrix; a counter number_cycles_of_life is initialized with 1.
3. A calculating function is applied to a new state of the automaton; we name the new state of the automaton chromosome_current; the counter number_cycles_of_life is increased with 1.
4. The sequential chromosome_current is compared with the values from the evolution matrix, from position 1 to position number_cycles_of_life-1 the sequential chromosome_current is compared to the values in the evolution matrix, from position 1 to position number_cycles_of_life-1
5. If an equality does not appear, the chromosome_current is memorized in the evolution matrix; and we came back to step 3
6. If an equality appears (the value already exists; that means that the chromosome_current represents the closure of the first cycle), the experiment stops, but (number_cycles_of_life -1) represents the performance of the chromosome; we archive the chromosome

Using this software device, we have generated over 500.000 chromosomes. The archiving of the results is done in a database. For each chromosome, we archive its initial state and its performance.

This represented the first stage of the project development. Normally, we shouldn't have done a detailed presentation of this stage. We have, nevertheless, a few reasons for having done it:

- The way the chromosomes evolution is studied doesn't change. As regards the crossover-obtained chromosomes, their evolution is studied the same way.

- With the definition inspired by G. Chaitin, we consider that an *automaton the longer life cycle has, the better approximates noise*. This definition will be used as fitness function throughout the genetic experiment. We will classify chromosomes using evolution as a fitness function: a chromosome is more performant if it has a longer life cycle.

([9], [10], [11], [14], [15], [16])

4.2 Crossover and Mutations

For *obtaining of new chromosomes*, we have applied the one-point crossover, as follows:

Parent 1

x1 x2 … x127 x128 | x129 x130 … x255 x256

Parent 2 ⇒

y1 y2 … y127 y128 | y129 y130 … y255 y256

Offspring 1

x1 x2 … x127 x128 | y129 y130 … y255 y256

Offspring 2

y1 y2 … y127 y128 | x129 x130 … x255 x256

(| was used for marking the place where the 'cut' has been made).

As a working mechanism, *mutation* appears too: with a certain frequency, a certain chromosome from the population is altered randomly

In our experiment, we have worked with a mutation rate of 1 $^{0}/_{00}$. ([5],[12])

5. The experiment

5.1 The Initial Gene pool

The first step: **initializing of the chromosomes** (selection of the initial population). We proceeded as follows: we had the database containing over 500.000 chromosomes. Within it, the chromosomes were recorded sequentially, the way they had appeared during the iterative generation. We next proceeded as follows:

1. we took 'slices' of 5000 chromosomes each, successively, as they appeared in the database.
2. we ordered them according to their performance.
3. after ordering the chromosomes, we retained for the initial population those classified within the first 16, which we will subsequently name 'favorites'.
4. from the rest of the initial population (the total of 5000 minus the favorites), we have chosen at random still 16 chromosomes, which we will subsequently call 'candidates' and which we have added to the favorites, thus obtaining the initial gene pool.

Thus, the 1st generation of the experiment was obtained.

5.2 Setting of the Parameters

1. We have used **256 cells chromosomes** (strings of length 256)
2. **The colonies have the length of 32** (they are formed of 32 chromosomes)
3. We have chosen the **initial generation** from the 5000 chromosomes **'gene pool'** as follows: those classified the first 16 (the favorites) plus other 16 (candidates), randomly chosen
4. With the chromosomes from the gene pool, we have made **32 crossings**: half with the favorites crossed with other chromosomes, and half with randomly chosen chromosomes.
5. We evaluated the 64 chromosomes (we calculated the length of their life cycle) and we classified them. **Those classified among the 32 first constitute a new generation.**v
6. The experiment **is iterated for 132 times.**

Our strategy aims at 'proliferating' of the favorites, or at their diffusion in the rest of the population. With the strategy employed here, favorites are always crossing: 32 from the 64 offspring are directly obtained from crossing them. The rest, randomly, from the idea that in a large search space, one can never know when comes next to a better solution.

6. Performances of the colonies

Any experiment with cellular automata starts from a certain solution to a given problem and tries to improve the respective solution. In our case, we want to obtain an improvement of the favorites.

In an experiment with cellular automata nothing is clear from the beginning. We let ourselves led by the intuition that *from a colony with better average performance (formed from the beginning from valuable individuals) chances of obtaining an increase of both the favorites and the colony are increased.*

Having these in mind, we have monitored throughout the experiment, several parameters: the initial average of the colonies and of the favorites, the final average of the colonies and of the favorites, and, in the idea of, perhaps, discovering a possible connection to the evolution of the colony, the ratio between the favorites and the colony, in the initial and in the final point.

6.1 Results

As a distribution of cases, the results are those in table 1

Table 1. Evolution of colonies

A	B	**C**	**D**	**E**	*F*	*G*	*H*	I	J
6-44	45	**44**	**0**	**44**	*1*	*1*	*0*	1	44
45-83	1	**1**	**1**	**0**	*0*	*0*	*0*	1	0
84-122	0								
123-161	2	**1**	**0**	**1**	*1*	*1*	*0*	1	1
162-200	3	**1**	**0**	**1**	*2*	*1*	*1*	1	2
201-239	2	**2**	**0**	**2**	*0*	*0*	*0*	0	2
240-278	44	**44**	**42**	**2**	*0*	*0*	*0*	42	2
279-317	1	**1**	**1**	**0**	*0*	*0*	*0*	1	0
318-356	0								
357-391	3	**3**	**3**	**0**	*0*	*0*	*0*	3	0

The contents of the columns is the following
A – the interval of initial averages of the favorites
B – TOTAL OF CASES
C – D – E THE AVERAGE OF FAVOURITES IS EQUAL, thus
C total cases, out of which:
D the colony increases
E the colony decreases
F – G – H THE AVERAGE OF FAVOURITES DECREASES, thus:
F total cases
G the colony increases
H the colony decreases
I TOTAL OF COLONIES INCREASING
J TOTAL OF COLONIES DECREASING

6.2 Comment on the results

We started from a few intuitions. Let us see the way they are confirmed (or not) after the experiments.

First of all, colonies: in 50 cases the average performance increased, and in 51 cases it decreased. That is, we have a balanced situation of scores.

With favorites, instead, the situation was the following: from all these experiments, the average of the favorites increased just in 4 cases. From these, in 3 cases, it was noticed an increase of the colony too, and in one case the colony remained stable.

The problem is that data obtained are not at all differentiated. More exactly, under similar conditions, the results are opposite.

As an example, let us examine in detail a few cases

In the value interval 6-44, from a total of 45 cases, in one case the colony is increasing, and in 44 the colony decreases (table 2.1). In bold type, the case of favorites increasing.

Table 2.1 Evolution of the experiment in the 6-44 interval

	The colony increases	The colony decreases Minimum/maximum
Initial average COLONY	**6.69**	5.25 – 7.53
Initial average FAVOURITES	**8.25**	6.00 – 10.00
Initial rapport FAVOURITES / COLONY	**1.23**	1.08 – 1.64

The increase of the favorites was from 8.25 to 102.25

In the value interval 123-161, out of a total of 2 cases, in one case the colony is increasing, and in one case the colony is decreasing (table 2.2). In bold type, the case where the favorites have increased.

Table 2.2 Evolution of the experiment in the 123-161 interval

	The colony increases	The colony decreases
Initial average COLONY	**39.06**	41.25
Initial average FAVOURITES	**133.60**	136.40
Initial rapport FAVOURITES / COLONY	**3.42**	3.31

The increase of the favorites was from 133.6 to 196.4

In the value interval 162 – 200, out of a total of 3 cases, in one case the colony is increasing, in one case the colony is decreasing, and in one case it doesn't change its performance. In bold type, the case where the favorites have increased.

Table 2.3 Evolution of the experiment in the 162-200 interval

	The colony increases	The colony decreases	The colony stays as such
Initial average COLONY	**46.41**	47.16	52.97
Initial average FAVOURITES	**165.90**	166.00	166.00
Initial rapport FAVOURITES / COLONY	**3.57**	3.52	3.70

The favorites' increase was from 165.9 to 229.10.

Conclusions

Several conclusions are drawn:

The first one, a rather sad conclusion: it cannot be made a direct connection between the rapport favorites-colony, initial and final.

Starting from what we have observed, we retain several constructive variants of resuming and continuing the experiments.

- We have cases where favorites have increased. Let us resume these cases, with the observation that, finally, we are not interested in the evolution of the colony, because we are hunting for the 'trophy' (the increase).
- Let us change the way of monitoring the experiment. More exactly, let us separate the pairs of parents having produced offspring with better performance. In fact, let us devise a database of pairs which, through crossing, have produced better results.
- Eiben A.E. et alii [5] makes a comparison between various crossover operators. We retain the idea of resuming our experiments, using several different crossover operators
- Eiben A.E. et alii [5] study also multi-parent reproduction strategies in genetic algorithms. A way of resuming the experiment could also be one where the offspring combine material from 4 parents, and not from two, like in our experiment.

References

[1] Borwein J., Borwein P., Girgensohn R., S.Parnes S.: *Experimental mathematics: A discussion*. Mathematical Intelligencer 18, 4 (May 1996), 12-18.

[2] Chaitin, G.: *On the length of programs for computing finite binary sequences*, Journal of ACM 13 (1966), p. 547-569

[3] Chaitin, G.: *On the length of programs for computing finite binary sequences: statistical considerations*, Journal of ACM 16 (1969), p. 145-159

[4] Chaitin, G.: *A theory of program size formally identical to information theory*, Journal of the ACM 22 (1975), pp. 329-340

[5] Davis, L: Handbook of genetic algorithms, Van Nostrand Reinhold, New York, 1991

[6]Eiben A.E., P-E. Raué P-E., Ruttkay Zs.: *Genetic Algorithms With Multi-Parent Recombination*, in Parallel Problem Solving from Nature III, Eds. Davidor Y., Schwefel H-P, Reinhard M., Springer, 1994, pp. 78—87

[7] Eiben A.E., Kemenade v. C.H.M., Kok J.N.: *Orgy in the computer: Multi-parent reproduction in genetic algorithms*, Advances in Artificial Life. Third International Conference on Artificial Life ed. Mor'an F Moreno A Merelo J. J and Chac'on P, volume 929 of Lecture Notes in Artificial Intelligence, pages 934-945, Springer, 1995

[8] Johnson D., "A Theoretician's Guide to the Experimental Analysis of Algorithms", AT&T Labs Research, 1996. Available from www.research.att.com/¸dsj/papers/exper.ps

[9] Mitchell, M., Hraber P. T., Crutchfield J. P.: *Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations*, Complex Systems, 7:89-130, 1993

[10] Mitchell, M., Hraber P. T., Crutchfield J. P.: *Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments*, Physica D, 75:361-391, 1994.

[11] Mitchell, M., Crutchfield J. P., Rajarshi Das: *Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work*, in Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96), Russian Academy of Sciences, 1996.

[12] Mitchell M.: An Introduction to Genetic Algorithms, A Bradford Book, The MIT Press, 1999

[13] Stefan, Gh.: *Looking for the lost noise*, CAS '98, *CAS '98 Proceedings*, Oct. 6 - 10, 1998, Sinaia, Romania. p.579 - 582.

[14] Wolfram, S.: *Cellular Automata as Simple Self-Organizing Systems*, Caltech preprint CALT-68-938 (1982)

[15] Wolfram, S.: *Cellular Automata*, Los Alamos Science, 9 (Fall 1983) 2-21

[16] Wolfram, S.: *Random Sequence Generation by Cellular Automata*, in Advances in Applied Mathematics, 7 (June 1986) 123-169