Research of Complex Forms in the Cellular Automata by Evolutionary Algorithms

Emmanuel Sapin, Olivier Bailleux, and Jean-Jacques Chabrier

Université de Bourgogne, 9 avenue A. Savary, B.P. 47870, 21078 Dijon Cedex, France
{olivier.bailleux, jjchab}@u-bourgogne.fr
emmanuelsapin@hotmail.com

Abstract. This paper presents an evolutionary approach for the search for new complex cellular automata. Two evolutionary algorithms are used: the first one discovers rules supporting gliders and periodic patterns, and the second one discovers glider guns in cellular automata. An automaton allowing us to simulate *AND* and *NOT* gates is discovered. The results are a step toward the general simulation of Boolean circuits by this automaton and show that the evolutionary approach is a promising technic for searching for cellular automata that support universal computation.

1 Introduction

Cellular automata are discrete systems in which a population of cells evolves from generation to generation on the basis of local transition rules. They can simulate simplified forms of life [1, 2] or physical systems with discrete time and space and local interactions [3–5].

Wolfram showed that one-dimensional cellular automata could present a large spectrum of dynamic behaviours. In "Universality and Complexity in Cellular Automata" [6], he introduces a classification of cellular automata, comparing their behaviour with that of some continuous dynamic systems. He specifies four classes of cellular automata on the basis of qualitative criteria. For all initial configurations, Class 1 automata evolve after a finite time to a homogeneous state where each cell has the same value. Class 2 automata generate simple structures where some stable or periodic forms survive. Class 3 automata's evolution leads, for most initial states, to chaotic forms. All other automata belong to Class 4. According to Wolfram, automata of Class 4 are good candidates for universal computation [6].

Among the 2-D automata with uniform neighborhoods, the only binary one currently identified as supporting universal computation is Life, which is in Class 4. Conway et al proved in [2] its ability to simulate a Turing machine, using gliders and glider guns. The gliders are periodic patterns which, when evolving alone, are reproduced identically after some shift in space. Glider guns emit a glider stream that carries information and creates logic gates through collisions. The identification of new automata able to simulate logic circuits is consequently a

promising lead in the search for new automata supporting universal computation. In this paper, we show how evolutionary algorithms can be used to look for automata accepting gliders and periodic patterns and for glider guns in these automata. In the following, we present a new automaton, discovered by evolutionary algorithm, simulating logic gates.

Section 2 describes the framework of our study and the used convention. Section 3 describes an evolutionary algorithm seeking new automata supporting gliders and periodic patterns. The search, by evolutionary algorithm, for glider guns is described in Section 4. Section 5 describes a discovered automaton simulating logic gates. Finally, in Section 6 we summarize our results and discuss directions for future research.

2 Framework

2.1 Cellular Automata

In this work, we explore only cellular automata with the following specifications:

-Cells have 2 possible values, called 0 and 1,

-they evolve in a 2D matrix, called *universe*, and

-the state of a cell at a generation i depends only on the states of itself and its eight neighbors at the generation *i-1*.

We call *context of a cell* the states of the cell and its eight neighbours [7]. Thus, a cell can have 512 different contexts. A transition rule is defined as a Boolean function that maps each of the 512 possible contexts to the value that will be taken by the concerned cell at the next generation. Therefore the underlying space of automata includes 2^{512} rules among which are the automata of Bays Space [8] including Life.

A context A can be defined by a matrix of which the elements are $A_{i,j}$, i and j included in {*-1, 0, 1*}. This is represented in Figure 1.

$A_{-1,1}$	$A_{0,1}$	$A_{1,1}$
$A_{-1,0}$	$A_{0,0}$	$A_{1,0}$
$A_{-1,-1}$	$A_{0,-1}$	$A_{1,-1}$

Fig. 1. Representation of the cells of a context

Two contexts A and B are *symmetrically equivalent* iff there exist n such that for every i and j, $A_{i,j}$ is equal to the n^{th} element of the list *(*$B_{i,j}$, $B_{-i,j}$, $B_{i,-j}$, $B_{-i,-j}$, $B_{j,-i}$, $B_{-j,i}$, $B_{j,i}$, $B_{-j,-i}$*)*. A group of symmetrically equivalent contexts is a set of contexts including all the symmetrically equivalent contexts of one context in the set. There are 102 groups of symmetrically equivalent contexts.

The simulation of logic circuits by Life uses a glider moving in any direction, so we choose to consider only *isotropic rules*. An isotropic rule is a rule in which

symmetrically equivalent contexts have the same associated value. In the representation of a rule, each group is identified by one of its elements (cf. Figure 3) and the associated value of each context is represented by the absence or the presence of a point at its right. This representation is used in the following.

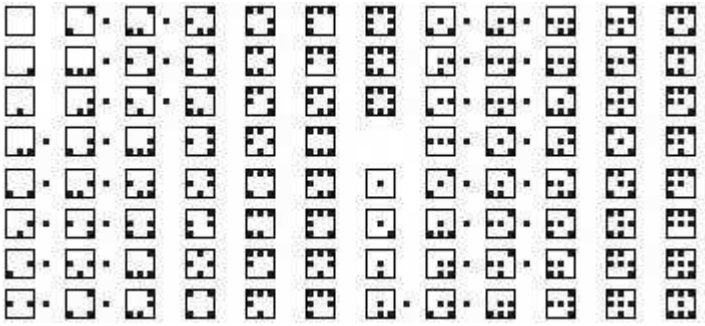

Fig. 2. Representation of a rule

We call *motif* a finite suit of Boolean matrix of dimension $k*k$. An automaton accepts a motif if said motif represents the successive states of the evolution of the first matrix by the transition rule of this automaton. The Life glider [9] is, for example, a motif accepted by Life. When every automaton supporting a motif maps the same values to a context c. We call c a *critical context* of this motif. For example, if a rule t supports a glider then all rules mapping the same values as t to the critical contexts of the glider support this glider.

3 A new rule

This section describes the utilization of an evolutionary algorithm for the search for new rules accepting gliders and periodic patterns.

3.1 The evolutionary algorithm

Encoding and operators Rules have been encoded by 512-bit strings in which all the bits of symmetrically equivalent contexts have the same value. The algorithm manages A rules. The bits of their transition function are randomly determined with a 0.5 probability. A mutation consists of modifying a randomly chosen bit, with the same weight for each of the 512 bits of a rule and likewise the bits of symmetrically equivalent contexts. Then, the isotropy of rules is kept. We implemented a simple crossover operator at a median point.

Fitness Function The computation of the fitness function is based on the evolution, during E transitions, of a "primordial soup", randomly generated in

a square of $C*C$ centered in a $U*U$ space with a density D. After this evolution the primordial soup is the object of the following test, inspired by Bays' test [8] to search gliders [10]: each group of connected cells is isolated in order to determine if it is a glider or a periodic pattern. In this way, the following algorithm tests each cell using the parameter R (maximum radius of a motif):

```
N=2
While (there is at least one living cell in a space of radius N
       from the central cell) AND (N<R)
  N=N+1
End while
```

Figure 3 gives an example in which some cells are in a space of radius 2 from the central cell (i.e. crossed by the large inner square) and neither in a space of radius 3. N will be equal to 3 at the end of the test, we call motif of radius 3.

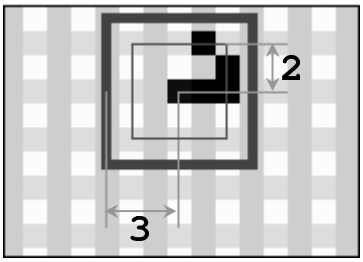

Fig. 3. Illustration of the search of a connected cells group.

Each group is inserted in an empty square of size T (size of the test space) and evolves during S generations (time of test). At each transition, the original pattern is sought in the test universe. Three cases can happen:

-The initial pattern has reappeared at its first location (this pattern is periodic).

-It has reappeared at another location (this pattern is a glider).

-It has not reappeared (it is then evolving).

The fitness function is evaluated as the multiplication of the number of occurrences of gliders by the number of occurrences of periodic patterns.

3.2 Evolutionary Algorithm

After the initialization of the A rules, the following cycle is iterated:

-The fitness function evaluates the rules.

-The X rules with the highest fitness function are kept.

- Y rules are created in the following manner, iterated Y times: choose one rule among the kept rules and mutate this rule.

-Z rules are created in the following manner, iterated Z times: choose two rules among the kept rules and apply crossover to these rules.

-A new population is obtained with the kept rule, the ones created by crossovers, and the ones created by mutations.

3.3 Parameters

Parameters of the fitness function Cellular automata that we study evolve in infinite space. In the simulation, space must be closed, thus we chose the size U in order not to close the evolution of the primordial soup within the universe's limits. As well as this, we chose the test space size T in order not to close the motif evolution test with the test space limits.

The size C and the density D of the primordial soup were chosen such that automata evolution promote the appearance of the highest number of periodical patterns and gliders. We assume that some size and density values that promote glider appearance in Life may also fit other automata.

If during the soup evolution in Life the average number of cells decreases too quickly, gliders cannot appear, and thus it must decrease as slowly as possible. We notice that the cell number stabilization during the soup evolution in Life show that only gliders and periodic forms survive, and thus new gliders can not appear.

In Life (cf. Figure 4) we look at, for several soups size with density 0.5, the ratio evolution of living cell number per the initial cell number. This ratio is evaluated during 80 generations. For a primordial soup with size 20, the curve decreases quickly and then stabilizes whereas for the other sizes the decrease is slower and the stabilization does not happen before 80 transitions. The curves for primordial soups of size 40, 60, and 80 merge. The soup size increases the time of evaluation of the fitness function, thus we chose to generate the primordial soup in a size 40 square.

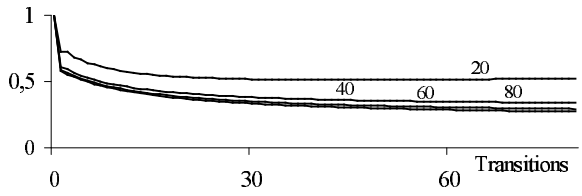

Fig. 4. Average over 100,000 evolutions of the ratio of the number of cells to the initial number of cells, during the evolution of primordial soups with density 0.5 by Life.

The evolution of primordial soups with size 40 by Life shows that the greatest number of cells must survive in order to increase the number of appearances of gliders and periodic patterns. Figure 5 gives the average number of living cells

during the evolution by Life of soups with size 40 and several densities. The decrease in the number of living cells is slowest for a 0.4 density, thus we chose this value for the D parameter.

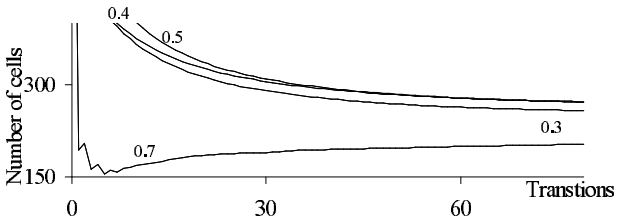

Fig. 5. Average number of cells during the evolution by Life of primordial soups of size 40 and of the indicated densities.

The evolution time of a primordial soup is linked to the time of evaluation of the fitness function, thus it is important this time is not too long. However, during the evolution of a soup a certain time is necessary for gliders and periodic patterns to be created. We counted the average number of gliders and periodic patterns (cf. Figure 6) that appeared during the evolution of 100,000 primordial soups by Life. We can see that a glider appears on average after 150 generations. Thus, we chose this value for the E parameter.

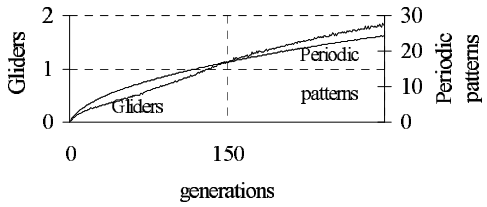

Fig. 6. Number of gliders (left axis) and periodic patterns (right axis) during the evolution by Life of primordial soups of size 40 and of density 0.4 evolving during the transitions from 0 to 300, for a maximum radius of 15 and a test time of 20.

The maximum radius of a motif must be determined in order to maximize the number of periodic patterns and gliders detected. A maximum radius that is too large slows down the calculation of the fitness function. In order to choose a radius, we generated several primordial soups evolving by randomly generated rules. During these evolutions, we counted how many gliders and periodic patterns had been detected with radius from 1 to 28. Figure 7 shows the percentage of motifs found for maximum radii from 1 to 28 (i.e., for a maximum radius of n only the motif with a radius less than or equal to n are counted). A maximum radius of 8 allows us to detect 85 percent of motifs; this value is chosen for the parameter R.

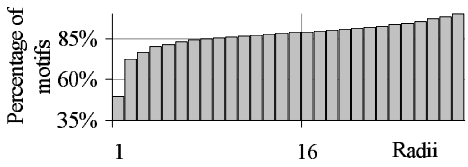

Fig. 7. Percentage of gliders and periodic patterns found for radius from 1 to 28 during the evolution of primordial soups of size 40 and of density 0.4 evolving until the transition 150 with randomly generated rules and a test time of 20.

The test time S must be determined in order to maximize the number of periodic patterns and gliders detected. Periodic patterns and gliders with a period greater than S will not be detected whereas a test time that is too long slowed down the calculation of the fitness function. In order to choose the test time, we generated several primordial soups evolving with randomly generated automata. During these evolutions, we counted how many gliders and periodic patterns were detected with test time from 1 to 20. Figure 8 shows the percentage of motifs found for test time from 1 to 20. We notice that a test time of 6 allows us to detect 97 percent of motifs; thus this value is chosen for the test time.

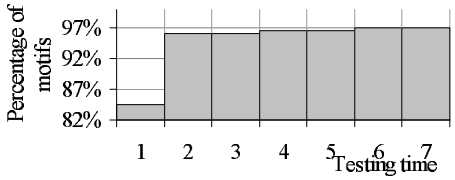

Fig. 8. Percentage of gliders and periodic patterns found for test time from 1 to 7 during the evolution of primordial soups of size 40 and density 0.4, evolving until the transition 150 with randomly generated rules, for a maximum radius of 8.

Parameters of the selection In order to determine the rate of mutation and crossover, different rates are tested by evolving our algorithm 100 times during 200 generations with a population of 50 automata. Figure 9 sums up the number of found automata for the following rates: 0, 20, 40, 60, and 80. The greatest number of automata is found for a rate of 80 for the mutation and a rate of 0 for the crossover. Thus, the simple crossover operator at a median point seems to be unsuitable for this algorithm.

The size A of the population must be chosen in order to optimise the number of automata found for a same number of automata evaluated by the fitness function. This result is presented is the next section.

Crossover \ Mutation	0%	20%	40%	60%	80%
0%	0	78	86	90	**98**
20%	41	67	79	93	
40%	30	69	88		
60%	28	87			
80%	10				

Fig. 9. Number of rules detected with 100 evolutions of our algorithm during 200 generations along with the indicated rates of crossover and mutation.

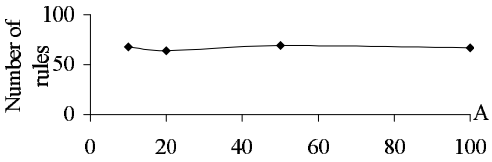

Fig. 10. Number of rules discovered by our algorithm for population sizes 10, 20, 50, and 100.

3.4 Experimental results

Figure 10 shows the number of discovered rules for 100 evolutions of our algorithm during 10, 20, 50, and 100 generations for population sizes 100, 50, 20, and 10, respectively. The number of discovered rules varies between 64 and 69 for different sizes of population. Thus this number is relatively constant for every population size tested but for a same number of evaluated automata. Each discovered automaton accepts one or more gliders. Some gliders appear in different automata. Figure 11 shows the gliders which appear the most often in the discovered rules with their appearance rate. Among the gliders discovered, these gliders have the least critical contexts. Their evolution goes through steps in which there are only three living cells that are the minimum (two cells define a line but not a direction).

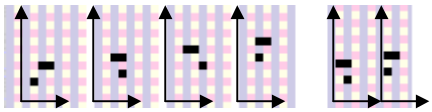

Fig. 11. Two gliders appear in 5.7 and 4.4 of the discovered rules by our algorithm and with, respectively, 11 and 9 critical contexts.

4 Research of new glider guns

In the discovered rules, using evolutionary algorithms with parameters and structure like the one in Section 3, we looked for glider guns that spontaneously emerged.

4.1 New evolutionary algorithm

The initial population is now made up of several occurrences of a rule discovered by the algorithm in the previous sections. A glider of this automaton is identified. The bits associated to its critical contexts are frozen (i.e. a non critical context is chosen for the mutation). Thus every discovered automaton accepts this glider. The fitness function is now evaluated by the division of the number of glider appearances by the total number of cells after the evolution time. During the detection of gliders, we look for glider guns. If the initial pattern has reappeared at its first location then it is removed and the leftover patterns are tested:

-If none rest then the pattern is periodic.

-If there is one or more gliders then a visual exam will determine if the initial pattern is a glider gun.

4.2 Results

Figure 12 shows the number of glider guns discovered for 100 evolutions of our algorithm during 50, 100, 250, and 500 generations for population sizes respectively 100, 50, 20, and 10.

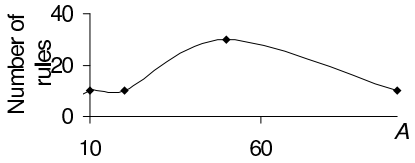

Fig. 12. Number of times when a glider gun is discovered by our algorithm for populations of sizes 10, 20, 50, and 100.

One of the discovered glider guns is shown figure 13.

Moreover, we found breeders which are patterns emitting one or more gliders but moving themselves.

5 Simulation of logic gates

In this section, we prove that one of the rules obtained using the previous algorithms, called R, can be used to simulate a logic gate. To the best of our

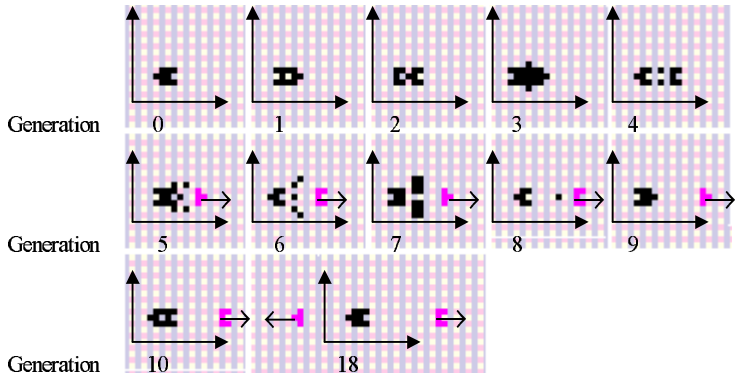

Fig. 13. A glider gun, discovered by evolutionary algorithm, with the gliders (lightly-coloured) and its direction shown by the arrows.

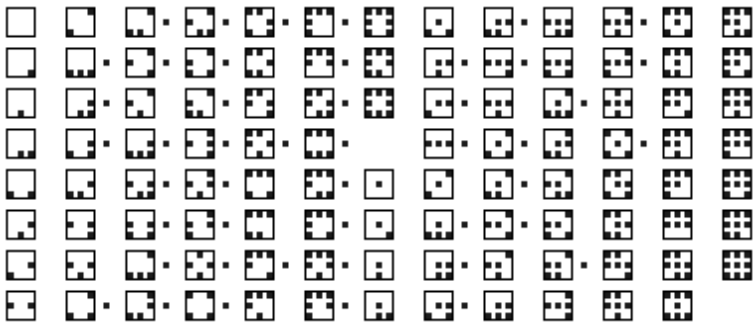

Fig. 14. The rule R.

knowledge, it is the first time that such a result is shown with a 2D automaton with two states and Moore's neighbor, different from Life.

The motifs used, in [9], to simulate a logic gate by Life are a glider gun, a glider, and an eater, i.e., a pattern that destroys the glider that crash it [11]. In R, we simulate a logic gate with these elements, Figure 15.

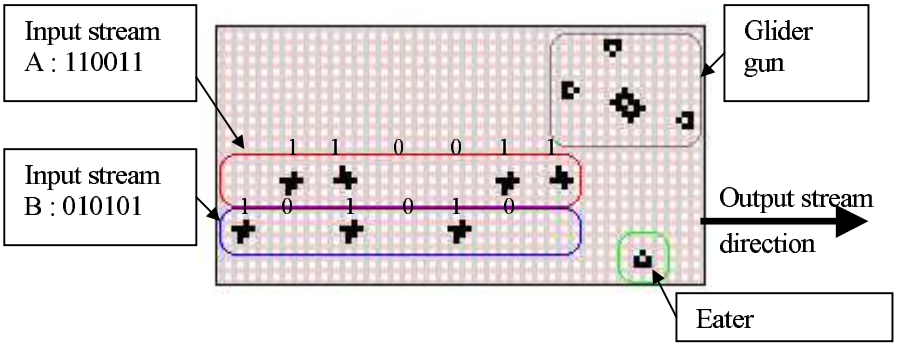

Fig. 15. A AND gate simulated by R.

In this simulation, the streams of gliders symbolize the streams of information (i.e., the absence of gliders symbolizes the value 0 and the presence of gliders the value 1). The glider stream created by the gun crashes the stream A. If one glider is present in the stream A, this collision destroys the two gliders else the glider emitted by the gun continue its run. Thus the resulting stream of this first collision is equal to $NOT(A)$, and this collision creates an NOT gate. This new stream crashes the stream B and two streams are the results of this second collision including one which is the result of the operation *"A AND B"* and another one which is destroyed by the eater. The synchronization and the position of the different elements are, of course, essential for the good working of the simulation. Thus the automaton R is able to simulate a AND and a NOT gate. This simulation of logic gate is a step toward the general simulation of logic circuits by R. This result is described in [12] using new patterns allowing us to duplicate glider streams [13] and to change their directions.

6 Synthesis and perspectives

We developed an evolutionary algorithm that discovers new automata supporting gliders, periodical patterns, and glider guns. We adopted an elitist strategy for which the best results are obtained without crossover. We plan to test other evolution strategies and to try several crossover operators. Using evolutionary algorithms, we identified a new automaton that is able to simulate logic gates AND and NOT. This simulation is a step toward the general simulation of logic

circuit. We plan to design an evolutionary algorithm able to discover automatically simulation of logic gates by automata. Later on, our aim will be the use of evolutionary algorithm for the search of new universal automata.

References

1. M. Gardner. "the fantastic combinaisons of john conway's new solitaire game " life ",". *In Scientific American*, 1970.
2. M. Gardner. "on cellular automata, self-reproduction, the garden of eden, and the game of life,". *In Scientific American*, 224:112–118, 1971.
3. C. Dytham and B. Shorrocks. "selection, patches and genetic variation: A cellular automata modeling drosophila populations,". *In Evolutionary Ecology*, 6:342–351, 1992.
4. I. R. Epstein. "spiral waves in chemistry and biology,". *In Science*, 252, 1991.
5. Ermentrout, G. Lotti, and L. Margara . "cellular automata approaches to biological modeling,". *In Journal of Theoretical Biology*, 60:97–133, 1993.
6. S. Wolfram. "universality and complexity in cellular automata,". *In Physica D*, 10:1–35, 1984.
7. E. Sapin, O. Bailleux, and J.J. Chabrier. "research of complexity in cellular automata through evolutionary algorithms,". *Submitted to publication.*
8. C. Bays. "candidates for the game of life in three dimensions,". *In Complex Systems*, 1:373–400, 1987.
9. E. Berlekamp, J.H Conway, and R.Guy. "winning ways for your mathematical plays,". *Academic press, New York*, 1970.
10. M. Magnier, C. Lattaud, and J-C. Heudin. "complexity classes in the two-dimensional life cellular automata subspace.,". *In Complex Systems*, 11, 1997.
11. E. Sapin, O. Bailleux, and J.J. Chabrier. "research of a cellular automaton simulating logic gates by evolutionary algorithms,". *In EuroGP03.Lecture Notes in Computer Science*, 2610:414–423, 2003.
12. E. Sapin, O. Bailleux, and J.J. Chabrier. "rapport technique lersia,universit de bourgogne,". *Publication interne*, 2003.
13. E. Sapin, O. Bailleux, and J.J. Chabrier. "a new approach of stream duplication in 2d cellular automata,". *Proceeding of SCI2003.*