Evolving Interesting Initial Conditions for Cellular Automata of the Game of Life Type

Manuel Alfonseca

Escuela Politécnica Superior
Universidad Autónoma de Madrid, 28049 Spain
manuel.alfonseca@uam.es

Francisco José Soler Gil

Universidad de Sevilla, Spain
and
Technische Universität Dortmund, Germany
soler@uni-bremen.de

The use of a genetic algorithm to obtain "interesting" initial conditions for cellular automata of the family of Conway's Game of Life is described in this paper. The conditions have been selected so as to maximize the number of gliders, R-pentominoes, and similar structures generated during the execution of the automata. Besides the original Game of Life rules, we have tested automata with similar rules, such as HighLife and B38S23, as well as mixed and time-dependent rules. We have concluded that the temporal invariance of the rules of these automata does not seem to be a requirement for the existence of the selected structures.

1. Introduction

Informally, a cellular automaton (CA) [1] is a set of finite deterministic automata distributed in discrete cells along a regular grid. The sets of states of its neighbors are inputs for each automaton, which means that the next state of each cell depends on its current state and on the states of all its neighbors; the neighborhood is usually the same all along the grid. CAs can be one-dimensional (1D) if the grid is a string of cells, bi-dimensional (2D) when the grid is a surface, or higher-dimensional. At every time step, also called a generation, each cell computes its new state by determining the states of cells in its neighborhood and applying transition rules to compute its new state. Every cell uses the same update rules and all cells are updated simultaneously.

When the grids are finite, boundary conditions become essential. They determine, for instance, which is the left neighbor of the leftmost cell. A typical boundary condition is called periodic (or cyclic):

1D rows are turned into circles (their extreme cells become adjacent to each other); 2D rectangular grids are turned into toroids (connecting the leftmost column to the rightmost column and the top row to the bottom row). Static (or closed) boundary conditions are also common, where extreme cells are assumed to be connected to permanent 0-state cells.

So far, CAs have proved very powerful in simulating many real-life applications and phenomena. It has been proved that some 1D and 2D CAs, such as the Game of Life (also known as Life), are computationally equivalent to the universal Turing machine [2].

The 2D CA called the Game of Life [3, 4] was designed by John Conway. It consists of a matrix of cells, where each cell may take one of two states: alive and dead (respectively represented by one and zero). Each cell has eight neighbors, according to a Moore neighborhood (in the eight main directions of the compass). The next state of a cell is determined by rule B3S23, which means that cells are born (go from dead to living state) if they have exactly three living neighbors, and survive if they have two or three living neighbors. In all other cases, a cell dies or remains dead.

Different variants of the Game of Life have been defined. HighLife, for instance, differs because its rule is B36S23 (i.e., a cell is also born if it has six living neighbors). Life-3-4 has rule B34S34 (cells are born or survive if they have three or four living neighbors). Seeds is a CA with rule B2S (a cell is born if it has exactly two living neighbors, but it never survives). We have also used variant B38S23.

The behavior of a CA depends on two different things: its definition (rules, neighborhood, size, and boundary condition) and its initial conditions (a matrix of initial states of cells in the grid). A given CA can be tested (executed) with different initial conditions. Depending on the CA definition and the initial conditions applied [5, 6], a CA can be classified into four broad categories: Class 1: ordered behavior (all cells take the same value); Class 2: periodic behavior; Class 3: random or chaotic behavior; Class 4: complex behavior. The first two are totally predictable. Random CAs are unpredictable. Somewhere in between, in the transition from periodic to chaotic, a complex, interesting behavior can occur.

Genetic algorithms have been used to evolve 1D [7–9] and 2D [10, 11] CAs to perform particular computational tasks, such as density classification or the production of predefined 2D and 3D shapes (form generation or morphogenesis). Sapin et al. [12] have used them to search through the rule space for CAs capable of sustaining logical AND gates. Most of these works have focused on the selection of rules, rather than initial conditions, as in our case.

Different modifications to the typical CA definition have been applied in the literature. In particular, rules may be probabilistic [13], fuzzy [14], or subject to changes, depending on the position of the cells [15–17]. Time-dependent rules have also been considered [18, 19], especially for music generation [20–22]. Somewhat similar to what we are proposing in this paper is the work by Chopard and Droz [23], whose CA applies two different rules in alternative generations.

This paper describes our experiments to evolve interesting initial conditions for 2D CAs of the Life type, and to study what happens to particular structures when the rules are changed. The 2D CAs we have used share the following definitions:

- The set of states, in our case {0,1}.
- The space size, in our case 60×60.
- The space boundary conditions, in our case a flat toroid.
- A neighborhood, in our case the Moore neighborhood (each cell has eight neighbors in the main eight directions of the compass).
- The CA rule, which describes the way in which the CA cell states change with time. The types of rules chosen are Life, HighLife, and B38S23, as well as mixed cases where rules are time-dependent and alternate in different ways.

Section 2 describes the genetic algorithm we have used to perform our experiments. Section 3 details the results of the experiments. Finally, Section 4 states our conclusions and future objectives.

2. Evolving Initial Conditions with a Genetic Algorithm

In our experiments, we first select a given CA definition (a rule, since all the other parameters of the CA are fixed). We call a set of initial conditions "interesting" when the evolution of the CA starting from them gives rise to one or more small CA structures, especially gliders (which make it possible to design logical gates, and thus provide Life with the capability for universal computation), but also R-pentominoes or exploders (see Figure 1). To compare different initial conditions and compute how interesting they are, the number of appearances of these structures (regardless of their type) is counted during a part of the CA's life. As gliders can remain unchanged during several life steps, they will usually count more than the other structures, which are ephemeral.

We have developed a genetic algorithm that discovers interesting initial conditions with the following parameters:

- Population size: 64 different random initial conditions for the chosen CA.
- Size of the original population random initial conditions: 30×30 centered on a 60×60 CA space. The remainder of the space is set at state 0 (dead cells).

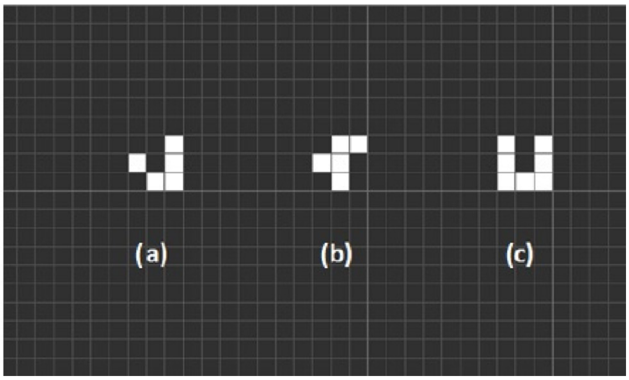

Figure 1. A few structures in the Game of Life: (a) glider, (b) R-pentomino, (c) exploder.

- Number of evolutionary steps: 400. We chose a fixed number of steps, rather than an evaluation of the algorithmic performance based on achieving a solution of the same quality, because this genetic algorithm results in a heavy-tail distribution. Therefore, reaching a given predefined fitness is not always possible, requiring an infinite number of steps. This happens frequently in this field, and the appropriate solution is restarting the algorithm after a fixed number of steps [24]. In this research, we set the restart number at 400. In other words, the algorithm is run for a fixed number of steps. (In the evolutionary algorithm jargon, steps are usually called generations, but here we call them steps to avoid misunderstandings with the generations of the CA.)
- The fitness score of each member of the population (each initial condition) is computed as the number of appearances of interesting structures during generations 40 to 54 during the execution of the CA. A glider that endures for several generations during this period counts for as many points as that number of generations. R-pentominoes and exploders, when they appear, are counted only at their first generation. Therefore gliders are much more strongly selected than the other structures. With these values, each complete execution of the genetic algorithm takes over half an hour, because the CA used by the algorithm must be run for each set of initial conditions in the population through generations 1 to 54 to compute their fitness. Reaching generation 100 would have approximately doubled the computing time. We thought that skipping the first 40 generations would allow the CAs to be sufficiently stabilized. In later experiments we tried different values for this parameter and found that testing generations 10 to 24 gives comparable (sometimes identical) results in about half the time.
- After each step in the evolutionary process, the eight individuals with the lowest scores are replaced by another eight, obtained from the eight with the highest scores that are paired randomly using the following genetic operations:

  - Recombination (crossover): the two parent initial conditions (raveled in row-first order) are split at a random point (the same for both) and recombined by appending the second part of each parent to the first part of the other. After this operation, if both parents are identical, the children will also be so.
  - Mutation: if both parents are identical, one random position of each child is always changed to a random state value. If both parents are different, this mutation is performed with a 10% probability. This is done to increase genetic variability when crossover generates children identical to their parents [25].

- After the indicated number of evolutionary steps, the program returns the initial condition that obtained the maximum score in the whole process. The total behavior of the chosen CA is then analyzed to find all the interesting structures it generates during its lifetime.

The number of individuals replaced in every step (eight, one-eighth of the population) was chosen because it provides the best balance between execution time, which is proportional to the number of individuals replaced, and fitness results. We tested a smaller number (two) where only the best two CAs in the population are allowed to remain, but no improvement was obtained in all the tests we made. We also tried higher numbers (16 and 32), where one-quarter or half the members are replaced by the children of the best fitted. They gave slightly better fitness improvements to the eight case, but required double or quadruple the computing time. Table 1 shows a comparison of the results obtained for eight and 16 replacements. In these experiments, the same set of seven random seeds was used for all CAs rules and genetic algorithm combinations.

Replace-ments	Rule	Final Fitness	Life Length	Gliders	Avg. Life	R-pent.	Expl-oders
16	Life=B3S23	45.86	578	106	33.49	23	71
8	Life=B3S23	37.14	776	118	37.03	25	124
16	HL=B36S23	39.29	386	41	53.63	5	24
8	HL=B36S23	27.29	415	43	22.00	18	25
16	L→HL→L→	41.71	522	65	64.34	13	27
8	L→HL→L→	34.14	429	63	47.25	14	26
16	L→SL→L→	38.86	930	170	31.22	44	137
8	L→SL→L→	42.57	497	68	52.03	14	54
16	All together	41.43	611	382	39.89	85	259
8	All together	35.29	529	292	40.51	71	229

Table 1. Comparisons of the average of seven different instances with different replacement numbers in the genetic algorithm (8 and 16).

Once the genetic algorithm has provided us with interesting initial conditions for a given CA, its function is finished. We then execute the CA starting with those initial conditions for as many generations as it keeps an interesting behavior, that is, until its behavior becomes periodic or static. We can also analyze what happens with these initial conditions if the rule of the CA is changed permanently or periodically.

3. Experimental Results

We have performed 20 experiments for each type of rule: Life (B3S23), HighLife (B36S23), B38S23, and two periodic mixed Life/HighLife and Life/B38S23 rules (the Life rule up to generation 25; the alternative HighLife/B38S23 rule up to generation 50; and so on, periodically). We have selected for maximum appearance of both gliders and R-pentominoes, although some exploders are also spontaneously generated.

Each experiment is considered to have ended when the CA configuration goes into a static situation, where the states of all the cells remain the same forever (not necessarily dead), or a periodic configuration, where the states of the cells oscillate with a certain period. Gliders (see Figure 2) are generated much more frequently than R-pentominoes. This is due to the fact that they retain the same appearance during a number of generations, which makes them easy to detect by the algorithm. R-pentominoes, on the other hand, are only easily detectable on the generation they first appear. Therefore we measure from gliders their number and their average duration; from R-pentominoes and exploders, we just measure their number of appearances.

This relative permanence, compared with the other objects, gives gliders the opportunity to display several interesting behaviors. Sometimes, for instance, an experiment generates very few gliders, but with a large duration; in other cases, gliders are generated and destroyed immediately. In a few experiments, two of the gliders collided and destroyed one another. In some cases, one or more gliders survive permanently, giving rise to a final periodic configuration with a period of 240 generations. (Since it takes a glider four generations to move a position diagonally, in a space of size 60×60, the period of a permanent glider is always 240.) We also show in Table 2 the total number of permanent gliders. In these experiments, the same set of 20 random seeds was used for all different types of CAs rules.

Table 2 summarizes the results of all the experiments as a function of the rule type.

Looking at Table 2 and the detailed results of the individual experiments, the following considerations can be made:

- Interesting behavior appears with all the rules (Life, HighLife, B38S23, and the mixed rules). The longest life and the largest number of gliders and exploders happened in one experiment with the Life rule, while the longest average life of the gliders appeared in one experiment with the HighLife rule. The largest number of R-pentominoes happened in a different experiment with the HighLife rule.

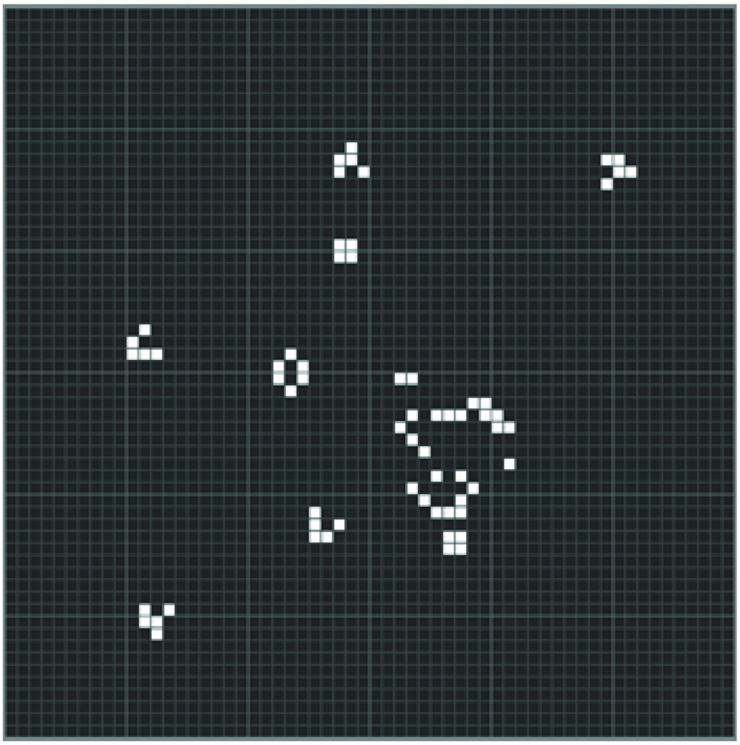

Figure 2. Five simultaneous gliders at one experiment.

Type of Rule	Avg. Life	Gliders	Perm. Gliders	Glider Life	R-pent.	Expl-oders	Int. Exp.
Life	717	307	12	38.24	69	263	12
HL	503	186	4	29.80	60	95	5
L→HL→L→	448	156	3	51.22	42	90	6
SL	743	339	6	36.16	75	223	13
L→SL→L→	548	259	7	39.95	53	159	9

Table 2. Summary of experiments as a function of rule type. The Life rule is B3S23. The HighLife rule is B36S23. SL represents rule B38S23. The last column shows the number of interesting experiments obtained.

- Sometimes the evolution experiments do not generate much interesting behavior (five or less gliders, few additional objects). This happens in eight out of the 20 Life experiments; 12 of the HighLife experiments; six of the B38S23 experiments; nine of the mixed L → HL → L → experiments; and eight of the mixed L → SL → L → experiments.
- Conversely, if we call an experiment especially interesting when 10 or more gliders are produced with an average life of over 10 generations, and at least another object appears (R-pentominoes or exploders), the numbers obtained are those shown in the last column in Table 2 (out of 20 experiments).
- This seems to indicate that the rules of Life and B38S23 are more prone to the appearance of interesting behavior than the rules of HighLife. The same conclusion is reached by comparing the numbers of interesting objects obtained for each type of rule (see Table 2).
- There are many different types of exploders (exploding structures), of which we have chosen to detect just two. Some of them are highly complex and interesting (see Figures 3 and 4). We have not selected these objects, but are counting them anyway because they appear quite frequently.

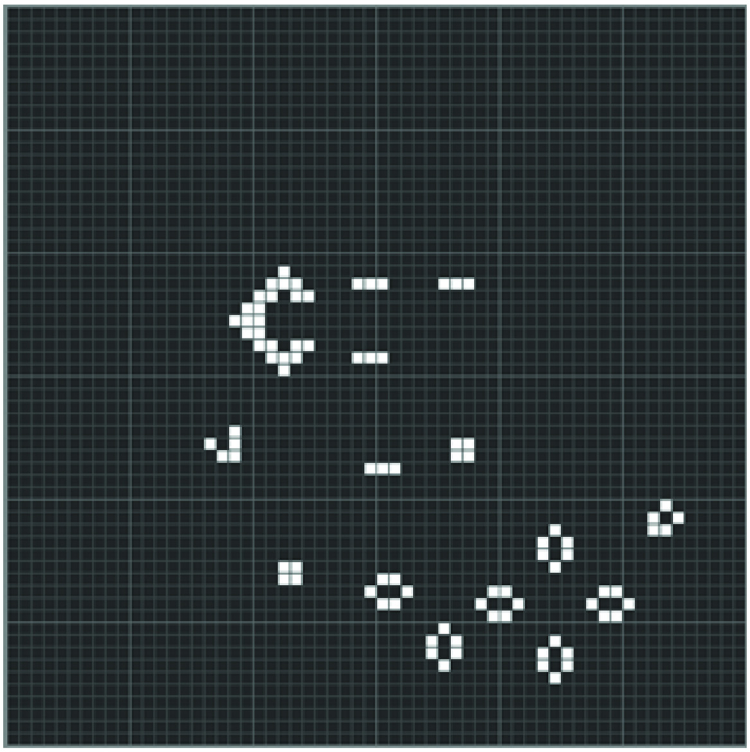

Figure 3. An exploder at generation 239 in one experiment. A glider is also visible.

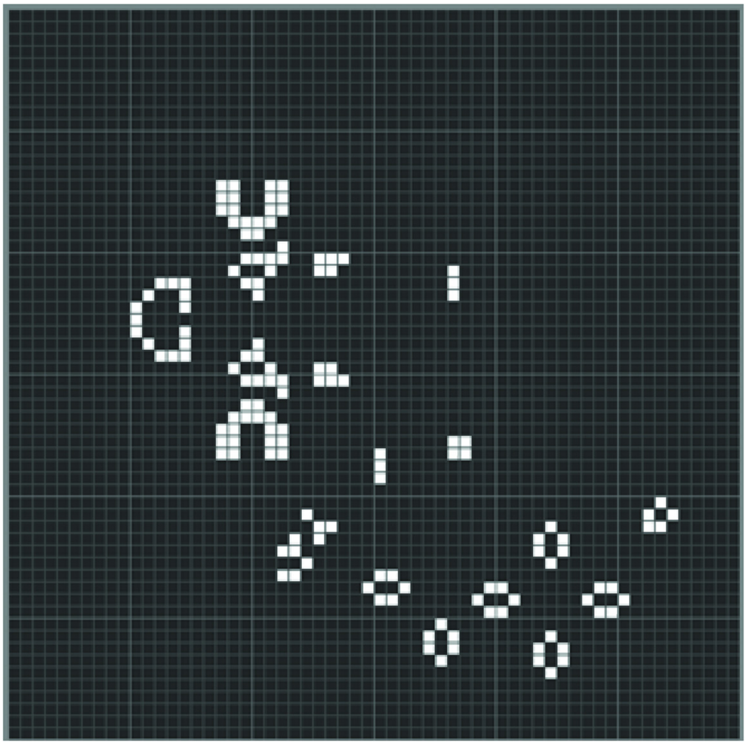

Figure 4. A further stage of the explosion at generation 268 in the same experiment. The glider has just collided against the small square at the bottom left.

3.1 Experiments Applying to a Cellular Automaton with a Given Rule an Initial Condition Evolved for a Cellular Automaton with a Different Rule

In the next set of experiments, we used the initial conditions evolved for Life with the HighLife rules and vice versa. We also tested the initial conditions evolved for Life with the B38S23 rule and vice versa. The results shown in Table 3 should be compared with those in Table 2. When the HighLife rule is used with the initial conditions evolved for Life, the results are much less interesting (shorter life length; very few gliders and similar objects appear; very few especially interesting experiments). In the opposite case, however, the situation is reversed: we get longer life lengths and a larger number of objects, with many more interesting experiments.

On the other hand, when the B38S23 rule is executed with the initial conditions selected for Life, only a slight decrease is observed. In the opposite case we also get a more interesting situation, with longer experiments and more objects (although a slightly smaller number of especially interesting experiments). Again, as in previous experiments, the HighLife rule seems to be less versatile than the Life rule. The B38S23 occupies an intermediate position, nearer to Life than to HighLife.

In the next set of experiments, we tried to find the effect of changing the rule (at generation 46) during the execution of some of the CAs used in the previous examples. So, if the automaton was generated using the rules of Life, after generation 46 the rules would change to HighLife and stay there for the remainder of its "life," and vice versa. Generation 46 was chosen because it is inside the training period for the genetic algorithm (generations 40 to 54). The results (averaged from 10 experiments) are shown in the first four rows in Table 4.

For comparison, the last three rows in Table 4 again show the results for the experiments described in Table 2, where the initial conditions evolved for one rule are executed on a CA with the same rule. Since Table 2 refers to 20 experiments versus the 10 in Table 4, those figures corresponding to total numbers of objects generated have been halved.

Evolved for	Executed for	Life Length	Gliders	Glider Life	R-pent.	Expl-oders	Int. Exp.
Life	HighLife	365	88	16.74	27	44	2
HighLife	Life	843	396	19.10	97	342	12
Life	B38S23	637	302	33.04	64	199	10
B38S23	Life	1384	732	18,39	205	551	11

Table 3. Experiments using a given rule for initial conditions evolved for a different rule.

Evolved for	Executed for	Life Length	Gliders	Glider Life	R-pent.	Expl-oders	Int. Exp.
Life	Life → HL	539	100	52.55	23	58	4.0
HL	HL → Life	767	212	29.63	37	131	4.0
Life	Life → SL	668	163	45,18	30	116	5.0
SL	SL → Life	620	140	37.16	18	82	5.0
Life	L→HL→L	543	128	45.36	28	91	5.0
Life	L→SL→L	504	104	54.44	16	69	5.0
Life	L→HL→L→	481	82	57.85	18	46	3.0
Life	L→SL→L→	646	190	35.44	41	107	4.0
Life	Life	717	153	38.24	34	131	6.0
HL	HL	503	93	29.80	30	47	2.5
SL	SL	743	169	36.16	37	111	6.5

Table 4. Experiments performed changing the rule during the execution. SL represents the rule B38S23.

From the observation of the results of these experiments, we can get the following conclusions:

- When the mixed rule of the form Life → HighLife was used with initial conditions evolved for Life, the CA displayed a less complex behavior (shortest life, less gliders and other interesting objects), compared to experiments where the rules of Life were always applied. A slightly more complex behavior was shown, however, compared to a HighLife CA with initial conditions evolved for HighLife, and a significantly more complex behavior than a HighLife CA provided with initial conditions evolved for Life.
- The mixed rules of the form HighLife → Life (with initial conditions evolved for HighLife) and Life → B38S23 (with initial conditions evolved for Life) generated a behavior about as complex than those where initial conditions evolved for Life or B38S23 were applied to a CA running with the same rules.
- The mixed rules of the form B38S23 → Life (with initial conditions evolved for B38S23) generated a behavior less complex than those where initial conditions evolved for Life or B38S23 were applied to a CA running with the same rules, but more complex than those described in the first bullet of this list.

This reinforces the conclusion derived from the first set of experiments: the rules of Life and B38S23 are more prone to the emergence of interesting behavior than the rules of HighLife.

In the next set of experiments, we tried to find out the effect of a very small change in the rules, rather than a permanent one. We started with 10 initial conditions evolved for CAs of the Life type and let them develop for 46 generations; then we changed the rules to HighLife or B38S23 (SL), executed them for four generations, and restored the rules to Life. The results are shown in rows five and six in Table 4. They show that mixed rules of the type L → HL → L are less disruptive than L → HL (leaving HL rules act for the remainder of the CA's life). The opposite effect happens when HighLife is replaced by B38S23.

In the last set of experiments, we tested the effect of a periodic change in the rules by executing 10 experiments with initial conditions evolved for Life on a CA with periodic rules (L → HL → L → and L → SL → L →). Rows seven and eight in Table 4 show the results. Again, the change to HighLife shows a more disruptive effect than the change to B38S23, which reaches numbers of objects comparable to the best, except for the number of interesting experiments. We can conclude that the latter periodic change in the rules, although having perceptible effects in each particular case, seems to diminish slightly the average complexity of the development.

4. Conclusions

In this paper we have found that it is possible to evolve interesting initial conditions for some cellular automata (CAs) of the Game of

Life type, which make the CAs generate structures such as gliders or R-pentominoes. Exploders are also generated in any case, but initial conditions for this case are not easy to select.

By testing different combinations where rules are changed during the execution of the CA, and through crossed application of initial conditions, we have found that the B36S23 (HighLife) rule is less prone to the emergence of interesting structures than the B3S23 (Life) and the B38S23 rules. We have also found (although this is not discussed in the paper) that CAs with very different rules, such as Life-3-4 and Seeds, do not give good results with our genetic algorithm.

In general, we can conclude that temporal invariance of the rules of the Game of Life type does not seem to be an essential requirement for the emergence and continued existence of potentially complex structures in this type of CA.

In the future, we intend to perform additional experiments with CAs of the Game of Life type for these or different objects, and also explore the effect of changing further the parameters and the genetic operations of the genetic algorithm.

References

[1] J. von Neumann, *Theory of Self-Reproducing Automata* (A. W. Burks, ed.), Urbana, IL: University of Illinois Press, 1966.

[2] P. Sarkar, “A Brief History of Cellular Automata,” *ACM Computing Surveys (CSUR)*, **32**(1), 2000 pp. 80–107. doi:10.1145/349194.349202.

[3] M. Gardner, “Mathematical Games: The Fantastic Combinations of John Conway’s New Solitaire Game ‘Life,’” *Scientific American*, **223**(4), 1970, pp. 120–123.

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

[5] S. Wolfram, “Universality and Complexity in Cellular Automata,” *Physica D*, **10**, 1984 pp. 1–35.

[6] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[7] M. Mitchell, P. T. Hraber, and J. P. Crutchfield, “Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations,” *Complex Systems*, **7**(2), 1993 pp. 89–130. http://www.complex-systems.com/pdf/07-2-1.pdf.

[8] M. Mitchell, P. T. Hraber, and J. P. Crutchfield, “Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments,” *Physica D: Nonlinear Phenomena*, **75**(1–3), 1994 pp. 361–391. doi:10.1016/0167-2789(94)90293-3.

[9] M. Mitchell, J. P. Crutchfield, and R. 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 (EvCA96)*, Moscow, Russia: Russian Academy of Sciences, 1996. http://www.cs.unibo.it/babaoglu/courses/cas06-07/resources/tutorials/Evolving_Cellular_Automata.pdf.

[10] F. Jiménez Morales, J. P. Crutchfield, and M. Mitchell, "Evolving Two-Dimensional Cellular Automata to Perform Density Classification: A Report on Work in Progress," *Parallel Computing*, **27**(5), 2001 pp. 571–585. doi:10.1016/S0167-8191(00)00078-8.

[11] A. Chavoya and Y. Duthen, "Using a Genetic Algorithm to Evolve Cellular Automata for 2D/3D Computational Development," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO06)*, New York: ACM, 2006. http://www.cs.york.ac.uk/rts/docs/GECCO_2006/docs/p231.pdf.

[12] E. Sapin and L. Bull, "Evolutionary Search for Cellular Automata Logic Gates with Collision-Based Computing," *Complex Systems*, **17**(4), 2008 pp. 321–338. http://www.complex-systems.com/pdf/17-4-1.pdf.

[13] S. A. Billings and Y. Yang, "Identification of Probabilistic Cellular Automata," *IEEE Transaction on Systems, Man, and Cybernetics, Part B*, **33**(2), 2003 pp. 225–236. doi:10.1109/TSMCB.2003.810437.

[14] P. Flocchini, F. Geurts, A. Mingarelli, and N. Santoro, "Convergence and Aperiodicity in Fuzzy Cellular Automata: Revisiting Rule 90," *Physica D: Nonlinear Phenomena*, **142**(1–2), 2000 pp. 20–28. doi:10.1016/S0167-2789(00)00052-X.

[15] A. K. Das, A. Ganguly, A. Dasgupta, S. Bhawmik, and P. P. Chaudhuri, "Efficient Characterisation of Cellular Automata," *IEE Proceedings-E: Computers and Digital Techniques*, **137**(1), 1990 pp. 81–87.

[16] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chattopadhyay, *Additive Cellular Automata: Theory and Applications*, Vol. 1, Los Alamitos, CA: IEEE Computer Society Press, 1997.

[17] P. Sarkar and R. Barua, "The Set of Reversible 90/150 Cellular Automata Is Regular," *Discrete Applied Mathematics*, **84**(1–3), 1998 pp. 199–213. doi:10.1016/S0166-218X(98)00004-3.

[18] A. Deutsch, *Cellular Automata and Biological Pattern Formation*, Boston: Birkhauser, 1999.

[19] N. Ganguly, B. K. Sikdar, A. Deutsch, G. Canright, and P. P. Chaudhuri, *A Survey on Cellular Automata*, Centre for High Performance Computing, Dresden University of Technology, 2003. http://www.cs.unibo.it/bison/publications/CAsurvey.pdf.

[20] P. Beyls, "The Musical Universe of Cellular Automata," in *Proceedings of the 1989 International Computer Music Conference* (T. Wells and D. Butler, eds.), 1989, pp. 34–41. http://quod.lib.umich.edu/cgi/p/pod/dod-idx?c=icmc;idno=bbp2372.1989.009.

[21] A. R. Brown, "Exploring Rhythmic Automata," *Lecture Notes in Computer Science*, **3449**, 2005 pp. 551–556. doi:10.1007/978-3-540-32003-6_57.

[22] D. Burraston, E. Edmonds, D. Livingston, and E. Reck Miranda, *Cellular Automata in MIDI Based Computer Music*, Ann Arbor, MI: University of Michigan, 2004.

[23] B. Chopard and M. Droz, *Cellular Automata Modelling of Physical Systems*, Cambridge: Cambridge University Press, 1998.

[24] M. Cebrián, A. Ortega, and M. Alfonseca, "Acceleration of a Procedure to Generate Fractal Curves of a Given Dimension through the Probabilistic Analysis of Execution Time," in *Intelligent Engineering Systems through Artificial Neural Networks*, Vol. 14, St. Louis, MO (C. H. Dagli, A. L. Buczak, D. L. Enke, M. J. Embrechts, and O. Ersoy, eds.), New York: ASME Press, 2004 pp. 265–270.
http://arantxa.ii.uam.es/~alfonsec/docs/annie.pdf.

[25] A. Ortega, A. A. Dalhoum, and M. Alfonseca, "Grammatical Evolution to Design Fractal Curves with a Given Dimension," *IBM Journal of Research and Development*, **47**(4), 2003, pp. 483–493
doi:10.1147/rd.474.0483.