ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2019.1.095

COMPARISON OF EVOLUTIONARY DEVELOPMENT OF CELLULAR AUTOMATA USING VARIOUS REPRESENTATIONS

Michal Bidlo

IT4Innovations Centre of Excellence, Faculty of Information Technology, Brno University of Technology, Czech Republic
bidlom@fit.vutbr.cz

Abstract

A comparative study is presented regarding the evolutionary design of complex multi-state cellular automata. In particular, two-dimensional cellular automata will be considered in combination with pattern development problem as a case study. Two techniques for the representation of transition functions for the cellular automata are proposed: a conventional table-based method and an advanced concept utilising conditionally matching rules. It will be shown that using a proper settings of Evolution Strategy, various working solutions can be obtained using both representations. Some observations from an analysis of resulting cellular automata will be presented which indicate that the behavior of the automata is totally different and depends on the representation applied. Specifically, the table representation exhibit a chaotic development during which a target pattern emerges at a moment. On the other hand, the conditional rules are able to achieve behavior that progressively constructs the target pattern which, in addition, represents a stable final state. Moreover, the latter method also exhibits significantly higher success rate which represents one of its advantages and proves an importance of systematic research in this area.

Keywords: *evolution strategy, cellular automaton, transition function, pattern development.*

Received: 11 April 2019
Accepted: 20 June 2019
Published: 24 June 2019

1 Introduction

Since the introduction of cellular automata (CA) by John von Neumann [23], researchers have dealt with the issue of efficient design of transition functions for the CA in order to achieve a given behavior (i.e. to perform a task by the CA). One of the main disadvantage of the CA concept is that designing transition rules for a specific task is not intuitive because it is not possible to use traditional programming techniques (e.g. known from imperative languages). This obstacle follows from the fact that the CA behavior represents an *emergent* process based on local interactions between cells. Therefore the design of cellular automata represents a difficult task whose complexity grows with increasing the number of cell states.

1.1 Overview of related work

Although many CA-based systems were successfully designed using analytical methods (for example, for the investigation into computational properties and construction of computing systems [14, 33, 4, 39, 36, 27], development of replicating structures [20, 12, 26, 37] or solving some mathematical operations ant other benchmarks in the cellular space [19, 13, 29, 38, 24]), the process of determining a suitable transition function for a given application represents a difficult task, especially due to an enormous growth of the solution space in dependence on the number of cell states. Therefore, the aim is to automate this process, using both deterministic algorithms and other heuristics or unconventional (non-deterministic) techniques, including evolutionary algorithms (EA).

Packard et al. [25, 28] were among the first who applied genetic algorithm (GA) [17] in order to adapt the transition rules for CA. Sapin et al. used an evolutionary approach to study the problem of discovering gliders in cellular array [32]. Sipper proposed a special technique, called Cellular Programming, for a parallel evolution of non-uniform CA, using a modified GA in each cell [34]. Another example represents the work of Breukelaar and Bäck who applied GA in order to evolve multi-dimensional uniform cellular automata to solve the density task and checker-board benchmarks [11]. Later, Sapin investigated the evolutionary discovery of glider guns in cellular automata [31]. Other soft-computing techniques have also been investigated in relation with CA. For example, Elmenreich et al. proposed an technique for calculating the transition function of CA using neural networks [15]. The goal was to train the network by means of Evolutionary Programming [16] in order to develop self-organising structures. In addition, various advanced concepts and modifications of cellular automata were investigated. For instance, Medernach studied a heterogeneous concept of cellular automata

whose cells utilise some advanced items like age, decay or genetic transfer using open-ended evolution to create an evolving ecosystem of competing cell colonies [22]. Bandini et al. dealt with effects in cellular automata that may be observed by introducing asynchronous update schemes [3]. In [18] a swarm intelligence algorithm, called Stochastic Diffusion Search, was applied in order to identify symmetry axes in patterns generated by cellular automata.

1.2 Motivation and goals

Despite the difficulties regarding the design of transition rules, cellular automata possess some interesting features that may be useful for modern areas like nanotechnology, molecular systems or fault-tolerant architectures. In particular, simple cell interactions, local interconnection, scalability or distributed computation represent typical aspects which are usually not present in conventional computing architectures. Therefore, it is important to continually extend our understanding of CA and their possibilities both on the application and elementary level.

For example, considering the recent progress in physical theory and information technology, advanced CA-based models have recently been studied involving principles from quantum theory [10, 21], nanoscale design [35, 1], implementation on the molecular level [30] or combination of some of these principles for the FPGA design [2].

The goal of this paper is to present a comparative study regarding the evolution of multi-state 2D cellular automata using various representations of transition functions and considering the pattern development problem from a seed as a case study. In particular, two evolutionary setups will be discussed considering a simple genetic algorithm and various setups of evolution strategies. The performance of each setup will be evaluated using two representations of the transition rules: the first using a traditional table-based method and the second using an advanced encoding called Conditionally Matching Rules. It will be shown that significant differences in the target CA behavior can be observed which depends on the representation applied. Moreover, successful experiments with the stable patterns development will be presented for the first time using the representation based on the conditionally matching rules.

2 Cellular Automata

The basic structure of a cellular automaton assumes a regular structure of *cells*, each of which at a given moment occurs in a *state* from a finite set of states. The behaviour (or *development*) of a CA will be considered as a *synchronous* update of all cells according to a *transition function* in discrete time steps. The transition function determines the next state of a cell depending on the combination of states its neighbourhood. For the purposes of this paper, (*uniform*) 2D CA will be considered with the shape of a square lattice and the cells sharing a *single* transition function. The cellular neighbourhood will be defined uniformly for each cell and will consist of a given cell and its immediate neighbors in the north, south, east and west direction (i.e. a 5-cell neighbourhood). Cyclic boundary conditions will be applied due to practical reasons following from the limitation of the CA size to a finite number of cells. The task of designing a CA consists of finding a suitable transition function in order to achieve a given CA behavior.

2.1 Table-based transition functions

Conventionally, the local transition function is represented by a *table* specifying *transition rules* for any possible combinations of states in its neighborhood. For the aforementioned 2D concept, the rules are of the form $s_N\ s_W\ s_C\ s_E\ s_S \rightarrow s_C^{new}$, where s_x denote the state of cell at position x in the neighborhood and s_C^{new} is a new state of the cell in the middle of the neighborhood in the next time step. For the purposes of this paper, a complete table will be considered which means that a new state value is specified for every possible combination of states in the neighborhood. Therefore, for a q-state CA with 5-neighborhood the transition function is represented by a linear array of q^5 integers, specifying the new cell states, with the left side of the rule being represented implicitly by the position in the array. However, the length of the array grows exponentially with increasing the number of states and thus the representation and design of the transition function becomes very difficult. It might be possible to specify a subset of rules (typically only those which modify the current state) but the problem is how to determine this subset for a given task especially for complex cellular automata. Therefore we chose the complete representation which can be represented in a straightforward manner.

2.2 Conditionally matching rules

Conditionally matching rules were introduced in [9] and subsequently their abilities were demonstrated on various benchmarks (e.g. [5, 6, 7]).

For the aforementioned CA setup, a conditional rule is defined as $(cnd_N\ s_N)\ (cnd_W\ s_W)\ (cnd_C\ s_C)\ (cnd_E\ s_E)\ (cnd_S\ s_S) \rightarrow s_C^{new}$, where cnd_x denote a condition operator ($=$, $\neq$, $\leq$ or $\geq$) and s_y represent a state value. Each CMR contains a pair *(a condition and a state value)* that corresponds to (is evaluated with respect to) a specific cell in the neighbourhood. A finite sequence of CMRs represents a transition function that, for example, contains a rule $(\neq 1)(\neq 2)(\leq 1)(\geq 2)(= 3) \rightarrow 2$. Let c_N, c_W, c_C, c_E, c_S be cells in states $2, 3, 0, 3, 3$ respectively, and a new state of c_C needs to be determined. The CMRs are evaluated sequentially until a rule is found whose all conditions are true with respect to the cell states in the given neighbourhood. According to the aforementioned rule, $c_N \neq 1$ is true as $2 \neq 1$, similarly $c_W \neq 2$ is true $(3 \neq 2)$ and the remaining conditions are also true. Therefore, this CMR is said to match, i.e. $s_C^{new} = 2$ on its right side will be the new state of c_C. If no matching rule is found, then the cell keeps its current state. Note that the CMR-based transition functions can be transformed to the appropriate table rules which preserves the original concept of CAs [7].

2.3 Case study: a (stable) pattern development

For the purposes of this paper, pattern development problem from a seed in 2D CA will be considered as a case study. In particular, the 9x9-cell Czech flag and 12x12-cell French flag will be developed. The initial CA state is considered as the middle cell of the CA in state 1 (representing the seed) with all other cells in state 0. The goal is to find a suitable transition function that will develop, in a finite number of steps, the given pattern from the initial state. Moreover, the target pattern is required to be *stable*, i.e. after its emergence no change in this CA state should occur within subsequent steps. For practical reasons, the CA of the size 4 cells larger in each dimension with the target pattern centered in the cellular array will be used. The target scenarios are illustrated in Fig. 1.

(a) 9x9-cell Czech flag pattern. (b) 12x12-cell French flag pattern.

Figure 1: Target patterns considered for the experiments.

3 Evolutionary Design of Cellular Automata

The objective of this paper is to perform a comparative study of the evolutionary design of cellular automata using the table-based representation and the CMR method. The experiments will be focused on complex multi-state 2D CA working with the numbers of states from 4 to 16. In the simplest case, there are in total $4^{4^5} \simeq 3.23 \times 10^{616}$ various transition functions for 4-state CA which makes the utilisation of EA justifiable.

3.1 Design of CA by means of genetic algorithm

One of the first extensive study dealing with the evolution of complex multi-state CA was proposed in [7], where a variant of simple genetic algorithm was applied. In particular, the GA worked with the population of 8 chromosomes, a tournament selection with base 4, a random mutation of 0–2 integers per chromosome and no crossover. This setup showed as reasonable for designing replicating structures in 2D CA in [7] using the CMR method. However, the GA-based approach failed completely in solving the benchmark proposed in Section 2.3. More precisely, from the settings of the number of CMRs from 20 to 80, the number of states from 2 to 16 and the population sizes 8, 16 and 32 individuals, the GA was able to find a single working solution only. This is probably caused by the fact that a stable pattern was required which was not the case in [7]. Another issue that contributed to failure of the GA may be the selection strategy applied that exhibit a low selection pressure for a good convergence of the algorithm towards working solutions. Therefore, more experiments followed regarding the design of a new EA that is able to fulfil this task not only with CMRs but (for the comparison purposes) with the table-based method as well. Therefore, further experiments were performed using some variants of (μ, λ)-evolution strategies (ES) which allow influencing the pressure by simply choosing a reasonable number

of parents μ and the number of offspring λ. According to some observations from the experiments regarding suitable setups of the ES, the evolution scheme applied will be referred to as Moderately Boosting Evolution Strategy and described in the next subsection.

3.2 Moderately Boosting Evolution Strategy

The evolutionary scheme applied in this paper is illustrated in Fig. 2. It is the (μ, λ)-Evolution Strategy, where μ is the number of parents in the population and λ is the number of offspring generated from parents, whereas $\lambda > \mu$. The initial population of μ individuals is generated randomly. During each evolutionary cycle, a parent is randomly chosen from the population in order to create an offspring by mutating the parent. The cycle repeats until λ offspring are created. Then, the offspring are evaluated using the fitness function (described in Section 3.3) and the best μ individuals are selected in order to replace the parents. The evolutionary cycle repeats this way until a working solution is found or a given number of generations is performed. Our experiments showed an advantage of this (μ, λ)-ES that enables evolution of complex cellular automata in a reasonable time and a good success rate. The concept of comma-selection strategy demonstrated a good ability to avoid getting stuck in local optima and the choice of the μ and λ values allows controlling the selection pressure in order to achieve a reasonable success rate. The details regarding the experimental setup will be stated in Section 4.

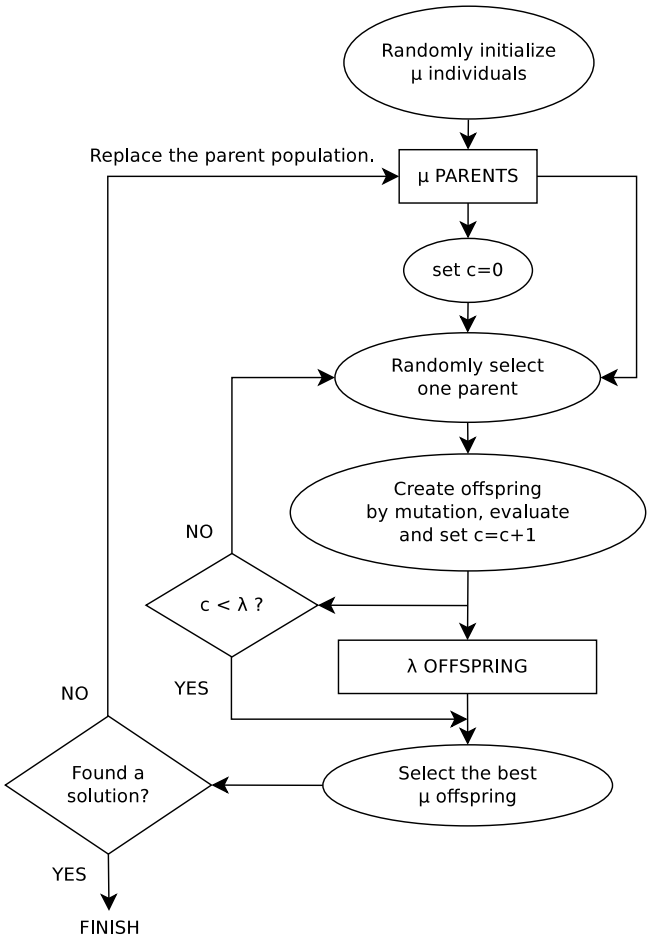

Figure 2: Evolutionary scheme of the (μ, λ) Evolutionary Strategy utilised for the experiments.

3.3 Evaluation of candidate solutions

Two approaches to the evaluation of solutions will be considered in this paper. For each type of experiment, a 2D CA consisting of $N \times N$ cells will be considered. In order to evaluate its behavior, the number of development steps $T = 22$ will be performed. The first approach considers a requirement of achieving a *stable* pattern that does not change during subsequent CA development. The second scenario requires achieving a given target pattern without the evaluation of its stability. In each scenario, the CA state obtained after each development step i is compared to the target pattern. For each CA state i a partial fitness value f_i is calculated as the number of cells in correct states. The calculation of the final fitness F differs for each scenario and is performed as follows.

Table 1: Success rates of evolutionary experiments using the table-based representation of transition functions for the development of patterns from Fig. 1 in CA without the requirement of pattern stability.

Target pattern	**9x9 Czech flag**							**12x12 French flag**						
#states	4	6	8	10	12	14	16	4	6	8	10	12	14	16
Evol. setup	**The num. of results out of 100 runs**													
(2, 8)-ES	0	5	1	8	7	1	0	0	0	1	4	2	4	0
(4, 16)-ES	0	2	0	3	4	0	0	0	1	1	0	3	0	0
(8, 32)-ES	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In the first scenario, the fitness is calculated as the maximum of the sum of two successive partial fitness f_{i-1}, f_i, i.e. $F = max(f_{i-1} + f_i), i = 1, 2, \ldots, T$. This ensures that the fitness is maximal for a state after the i-th step, that fully corresponds to the target pattern, and has not changed from the previous state, i.e. the pattern is stable. Therefore the objective is to maximize the fitness F whereas the fitness of a fully working solution $F_{max} = 2 * N^2$.

In the second scenario, the final fitness is calculated as the maximum out of the partial fitness values f_i for $i = 1, 2, \ldots, T$. In this case the requirement is to achieve a state in at least one of the development steps that fully corresponds to the target pattern. Hence the fitness of a fully working solution $F_{max} = N^2$.

4 Experimental Results

The goal of the experiments was to compare the ability of the evolution strategies to find suitable transition functions for 2D CA to solve the pattern development from a seed as described in Section 2.3. Specifically, (2, 8)-ES, (4, 16)-ES and (8, 32)-ES were applied to design CA working with 4, 6, 8, 10, 12, 14 and 16 cell states. In order to perform an objective comparison and ensure the same number of fitness evaluations in each ES variant, the maximum generation limit was set to 2 million generations in (2, 8)-ES, 1 million generations in (8, 16)-ES and 500 thousand generations in (8, 32)-ES. In each experimental setup, 100 independent evolutionary runs were conducted using the table-based transition function representation and conditionally matching rules.

The success rates of experiments working with the table representation are summarized in Table 1. As evident, two out of three ES setups were able to find some working solutions, although the success rates are under 10%. It is also important to state that the table-based experiments were conducted with the second fitness scenario, i.e. the stability of the pattern was not requires. The reason for this is that no working solution was found for stable patterns.

The success rates for the experiments working with conditionally matching rules are summarized in Table 2. In each ES setup, the number of CMRs were considered from 20 to 80 with step 10. This set of experiments considered the first fitness scenario in which stable patterns were required. As can be seen, the CMR representation provides solutions mostly with significantly higher success rates in comparison with the table method which is particularly true for the 9x9 Czech flag pattern. This can be viewed as a strong ability of the CMR representation to evolve complex CA especially in combination with the requirement of stable patterns. These experiments probably show for the first time a successful evolutionary design of complex multi-state CA for the development of stable patterns composed of more than three cell states. The comparison of resulting CA behavior has shown significant differences between the two proposed representations which will be described in the following paragraphs. For this purpose we show the more complex case study – the 12x12 French flag development.

A sample of evolved CA behavior using the table representation is shown in Fig. 3a. In fact, the table-based solution does not perform a systematic development. Note that a simpler fitness scenario was considered in this case, which does not require achieving stable patterns, because these experiments using the table method were not successful. As Fig. 3a shows, the initial seed "grows" into a chaotic state from which, at some time, the target pattern emerges. However, it could be shown that if the CA developed further, the pattern would be destroyed and never reconstructed again. This feature was observed in all results obtained by the evolution of table representation. Moreover, in every case the development of the target pattern is strictly dependent on the size of the CA used during evolution, i.e. these transition functions are not robust.

The behavior evolved using the CMR method is shown in Fig. 3b. As evident, the solution obtained from the CMRs *progressively* constructs the pattern which then remains stable if the CA develops further. Note that such a distinguished CA behavior was observed in *all* results regarding the two representations. Moreover, some CMR solution (even if not all) works in CA of arbitrary size, i.e. the creation of the given pattern is not dependent on the CA used during evolution in which the boundary conditions might influence the result. The CMR approach also allowed for achieving stable patterns (as required explicitly by the fitness evaluation). None of the two aforementioned features was achieved in case of the table representation. Nevertheless, the solution

Table 2: Success rates of evolutionary experiments using the CMR-based representation of transition functions for the development of patterns from Fig. 1 in CA with the requirement of pattern stability.

Target pattern	**9x9 Czech flag**							**12x12 French flag**						
#states	4	6	8	10	12	14	16	4	6	8	10	12	14	16
Evol. setup	**The num. of results out of 100 runs**													
(2, 8)-ES, 40 CMRs	0	0	0	0	0	0	1	0	0	0	0	0	0	0
(2, 8)-ES, 50 CMRs	0	1	0	2	5	3	2	0	0	0	0	0	0	0
(2, 8)-ES, 60 CMRs	0	11	15	18	14	14	15	0	0	0	0	0	0	0
(2, 8)-ES, 70 CMRs	0	19	21	34	39	43	36	0	0	0	0	0	0	0
(2, 8)-ES, 80 CMRs	0	22	45	42	47	49	39	0	0	0	0	0	0	1
(4, 16)-ES, 30 CMRs	0	5	5	5	10	5	7	0	0	0	0	0	0	0
(4, 16)-ES, 40 CMRs	0	14	33	16	29	22	26	0	0	0	0	2	2	0
(4, 16)-ES, 50 CMRs	1	12	30	22	26	27	23	0	0	2	3	2	2	7
(4, 16)-ES, 60 CMRs	1	17	18	16	19	28	18	0	0	5	6	6	7	4
(4, 16)-ES, 70 CMRs	0	9	18	19	16	19	13	0	0	3	4	5	2	3
(4, 16)-ES, 80 CMRs	0	8	14	10	17	10	11	0	0	3	3	3	2	5
(8, 32)-ES, 20 CMRs	0	0	0	4	1	0	1	0	0	0	0	0	0	0
(8, 32)-ES, 30 CMRs	0	4	6	12	7	11	14	0	0	0	0	0	1	0
(8, 32)-ES, 40 CMRs	0	6	6	8	12	9	9	0	0	1	2	2	1	1
(8, 32)-ES, 50 CMRs	0	5	11	13	8	9	9	0	0	3	3	4	1	2
(8, 32)-ES, 60 CMRs	0	5	5	8	9	4	10	0	0	0	1	1	1	4
(8, 32)-ES, 70 CMRs	0	3	3	5	6	7	7	0	0	0	0	1	1	1
(8, 32)-ES, 80 CMRs	0	1	4	3	3	6	5	0	0	0	0	1	1	0

evolved by means of the CMR representation are fully convertible to the appropriate table rules, i.e. there is no difference in the CA concept applied.

(a) A sample of 12x12 French flag development according to a transition function evolved by means of the table method.

(b) A sample of 12x12 French flag development according to a transition function evolved by means of the CMR method.

Figure 3: Comparison of the CA development designed by means of the table and CMR representation.

Although all case studies provided some successful results, there can (naturally) be observed differences in difficulties to find a working solution (which can be expressed in terms of success rates presented in Table 1 and 2). The most difficult showed to be the table-based development of 12x12 French flag and the same problem solved by the CMRs with the additional requirement of the pattern stability. On the other hand, the CMR-based development of 9x9 Czech flag was relatively easy for the evolution as observable from success rates which are several times higher against the other case studies. This shows (together with the different ways of CA behavior observed) that the representation of transition functions plays a key role to design a given CA. As evident from the results, the advanced method using the conditionally matching rules can be very promising for designing complex CA. Although some analyses have been performed so far regarding transition functions evolved by the two representations, there is still no evidence for what the main cause can be which makes the resulting CA behavior so different. One of the possibilities may be the fact that a single CMR may implicitly represent several different table rules which helps the evolution in searching the solutions. Another aspect can lie in the condition functions that allow more robust "inspection" of the cellular neighborhood instead of exactly matching the cell states. However, the possibility to express a single transition rule remains preserved.

In total, there are over a thousand different solutions obtained from the experiments performed for this paper. In order to support research in this area, all the results have been provided in the table representation together with an interactive Python-based CA simulator and are available from [8].

5 Conclusions

A comparative study was presented regarding the evolutionary design of complex multi-state cellular automata. In particular, two-dimensional cellular automata were considered in combination with pattern development problem as a case study. Two techniques for the representation of transition functions for the cellular automata were considered: a conventional table-based method and an advanced concept utilising conditionally matching rules. It was shown that using a proper settings of Evolution Strategy, various working solutions could be obtained using both representations. Some observations from an analysis of resulting cellular automata showed that the behavior of the automata is totally different and depends on the representation applied. Specifically, the table representation exhibited rather chaotic development during which a target pattern emerged at a specific moment but further is destroyed and never reconstructed again. On the other hand, the conditional rules achieved a progressive construction of the target pattern which, in addition, remains stable during further CA development. Moreover, this approach also exhibits significantly higher success rate which represents one of its advantages and proves an importance of further research in this area.

Despite many various results obtained in each of the case studies, there are some open questions that may be important for better understanding of the CA design and its behavior. For example, exact relations between the evolved table representation and the table representation obtained from the evolved conditionally matching rules is not yet known although both utilise the same CA concept. The CMR method indicates that simpler and more robust CA can be evolved in comparison with the table representation. Other question may arise the fact that the table based method always considers a complete set of transition rules which may be limiting for more efficient evolution. Are there more efficient approaches that could benefit from simplicity of the table method and provide better results? How to choose a subset of rules which are really needed to solve a given problem? These issues represent topics for the future research.

Acknowledgement: This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic INTER-COST project Advanced Methods of Nature-Inspired Optimisation and HPC Implementation for the Real-Life Applications – LTC18053, and Large Infrastructures for Research, Experimental Development and Innovations project of the IT4Innovations National Supercomputing Center – LM2015070.

References

[1] Bahar, A. N., Waheed, S., and Hossain, N. 2015. A new approach of presenting reversible logic gate in nanoscale. *SpringerOpen Journal - Electronics and Electrical Engineering* 4. DOI: 10.1186/s40064-015-0928-4

[2] Balijepalli, H. and Niamat, M. 2012. Design of a nanoscale quantum-dot cellular automata configurable logic block for fpgas. In *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, pp. 622–625.

[3] Bandini, S., Bonomi, A., and Vizzari, G. 2012. An analysis of different types and effects of asynchronicity in cellular automata update schemes. *Natural Computing* 11, 2, pp. 277–287.

[4] Berlekamp, E. R., Conway, J. H. and Guy, R. K. 2004. *Winning Ways for Your Mathematical Plays*, 2nd Ed., Volume 4. A K Peters/CRC Press, USA.

[5] Bidlo, M. 2015. Investigation of replicating tiles in cellular automata designed by evolution using conditionally matching rules. In *2015 IEEE International Conference on Evolvable Systems (ICES), Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 1506–1513.

[6] Bidlo, M. 2016. Evolution of generic square calculations in cellular automata. In *Proceedings of the 8th International Joint Conference on Computational Intelligence - Volume 3: ECTA*. SciTePress - Science and Technology Publications, pp. 94–102.

[7] Bidlo, M. 2016. On routine evolution of complex cellular automata. *IEEE Transactions on Evolutionary Computation* 20, 5, pp. 742–754.

[8] Bidlo, M. 2019. Results of evolutionary development of cellular automata using various representations with interactive simulator. URL https://github.com/bidlom/Mendel2019-ES

[9] Bidlo, M. and Vasicek, Z. 2013. Evolution of cellular automata with conditionally matching rules. In *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*. IEEE, pp. 1178–1185.

[10] Bisio, A., D'Ariano, G., Perinotti, P., and Tosini, A. 2015. Free quantum field theory from quantum cellular automata. *Foundations of Physics* 45, 10, pp. 1137–1152.

[11] Breukelaar, R. and Bäck, T. 2005. Using a genetic algorithm to evolve behavior in multi dimensional cellular automata. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO 2005*. ACM, pp. 107–114.

[12] Byl, J. 1989. Self-reproduction in small cellular automata. *Physica D: Nonlinear Phenomena* 34, 1–2, pp. 295–299.

[13] Capcarrere, M. S., Sipper, M., and Tomassini, M. 1996. Two-state, $r = 1$ cellular automaton that classifies density. *Physical Review Letters* 77, pp. 4969–4971.
[14] Codd, E. F. 1968. *Cellular Automata.* Academic Press, New York, USA.
[15] Elmenreich, W. and Fehérvári, I. 2011. Evolving self-organizing cellular automata based on neural network genotypes. In *Proc. of the 5th International Conference on Self-organizing Systems.* Springer, pp. 16–25.
[16] Fogel, L. J., Owens, A. J., and Walsh, M. J. 1966. *Artificial Intelligence through Simulated Evolution.* Wiley, New York, USA.
[17] Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, USA.
[18] Javaheri Javid, M. A., al Rifaie, M. M., and Zimmer, R. 2014. Detecting symmetry in cellular automata generated patterns using swarm intelligence. In: *Theory and Practice of Natural Computing, Lecture Notes in Computer Science.* Springer, vol. 8890, pp. 83–94.
[19] Land, M. and Belew, R. K. 1995. No perfect two-state cellular automata for density classification exists. *Physical Review Letters* 74, pp. 5148–5150.
[20] Langton, C. G. 1984. Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena* 10, 1–2, pp. 135–144.
[21] Mardiris, V., Sirakoulis, G., and Karafyllidis, I. 2015. Automated design architecture for 1-d cellular automata using quantum cellular automata. *IEEE Transactions on Computers* 64, 9, pp. 2476–2489.
[22] Medernach, D., Kowaliw, T., Ryan, C., and Doursat, R. 2013. Long-term evolutionary dynamics in heterogeneous cellular automata. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation.* ACM, pp. 231–238.
[23] von Neumann, J. 1966. *The Theory of Self-Reproducing Automata.* University of Illinois Press, USA.
[24] Ninagawa, S. 2013. Solving the parity problem with rule 60 in array size of the power of two. *Journal of Cellular Automata* 8, 3–4, pp. 189–203.
[25] Packard, N. H. 1988. Adaptation toward the edge of chaos. In *Dynamic Patterns in Complex Systems.* World Scientific, pp. 293–301.
[26] Reggia, J. A., Armentrout, S. L., Chou, H. H., and Peng, Y. 1993. Simple systems that exhibit self-directed replication. *Science* 259, 5099, pp. 1282–1287.
[27] Rendell, P. 2013. A fully universal turing machine in Conway's game of life. *Journal of Cellular Automata* 9, 1–2, pp. 19–358.
[28] Richards, F. C., Meyer, T. P., and Packard, N. H. 1990. Extracting cellular automaton rules directly from experimental data. *Physica D* 45, 1–3, pp. 189–202.
[29] Sahoo, S., Choudhury, P. P., Pal, A., and Nayak, B. K. 2014. Solutions on 1-d and 2-d density classification problem using programmable cellular automata. *Journal of Cellular Automata* 9, 1, pp. 59–88.
[30] Sahu, S., Oono, H., Ghosh, S., Bandyopadhyay, A., Fujita, D., Peper, F., Isokawa, T., and Pati, R. 2010. Molecular implementations of cellular automata. In *Cellular Automata for Research and Industry, Lecture Notes in Computer Science.* Springer, Vol. 6350, pp. 650–659.
[31] Sapin, E. 2010. Gliders and glider guns discovery in cellular automata. In: *Game of Life Cellular Automata.* Springer, pp. 135–165.
[32] Sapin, E., Bailleux, O., and Chabrier, J. 1997. Research of complexity in cellular automata through evolutionary algorithms. *Complex Systems* 17, 3, pp. 231–241.
[33] Sipper, M. 1995. Quasi-uniform computation-universal cellular cutomata. In *Advances in Artificial Life, ECAL 1995, Lecture Notes in Computer Science.* Springer, Vol. 929, pp. 544–554.
[34] Sipper, M. 1997. *Evolution of Parallel Cellular Machines – The Cellular Programming Approach*, Lecture Notes in Computer Science, Vol. 1194. Springer, Berlin, Germany.
[35] Sridharan, K. and Pudi, V. 2015. *Design of Arithmetic Circuits in Quantum Dot Cellular Automata Nanotechnology.* Springer International Publishing, Switzerland.
[36] Stefano, G. D. and Navarra, A. 2012. Scintillae: How to approach computing systems by means of cellular automata. In *Cellular Automata for Research and Industry*, Lecture Notes in Computer Science. Springer, Vol. 7495, pp. 534–543.
[37] Tempesti, G. 1995. A new self-reproducing cellular automaton capable of construction and computation. In *Advances in Artificial Life, Proc. 3rd European Conference on Artificial Life, Lecture Notes in Artificial Intelligence.*Springer, Vol. 929, pp. 555–563.
[38] Wolfram, S. 2002. *A New Kind of Science.* Wolfram Media, Champaign IL, USA.
[39] Yunes, J. B. 2010. Achieving universal computations on one-dimensional cellular automata. In *Cellular Automata for Research and Industry, Lecture Notes in Computer Science.* Springer, Volume 6350, pp. 660–669.