On Routine Evolution of Complex Cellular Automata

Michal Bidlo

***Abstract*—The paper deals with a special technique, called conditionally matching rules, for the representation of transition functions of cellular automata and its application to the evolutionary design of complex multi-state cellular automata. The problem of designing replicating loops in two-dimensional cellular automata and the square calculation in one-dimensional cellular automata will be treated as case studies. It will be shown that the evolutionary algorithm in combination with conditionally matching rules is able to successfully solve these tasks and provide some innovative results compared to existing solutions. In particular, a novel replication scheme will be presented that exhibits a higher replication speed in comparison with the existing replicating loops. As regards the square calculation, some results have been obtained that allow a substantial reduction of the number of steps of the cellular automaton against the currently known solution. The utilisation of the conditionally matching rules in the proposed experiments represents the first case of a successful automatic evolutionary design of complex cellular automata for solving non-trivial problems in which the existing conventional approaches have failed.**

***Index Terms*—Cellular automaton, evolutionary algorithm, square calculation, replicating loop.**

I. Introduction

SINCE the introduction of cellular automata (CA) in [1], researchers have dealt with the problem of how to effectively design a cellular automaton (and its transition function in particular) to solve a given task. Although many successful applications of cellular automata have so far been presented in various domains (e.g. concerning artificial life [2][3], nanocomputing [4], image processing [5][6], molecular simulations [7] or even the utilisation of DNA molecules to constructing nano-scale CA-based computing devices [8] including some applications of this concept [9]), the process of designing suitable rules to solve specific problems in CA still represents a difficult task.

The paper deals with the evolutionary design of cellular automata using a special representation technique referred to as Conditionally Matching Rules (CMRs). 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 the cell states according to a *transition function* in discrete time steps. It will also been assumed that the states are discrete (integer) values. The transition function determines the next state of a cell depending on the combination of states in a cellular neighbourhood that includes the cell to be updated and its neighbours. There are two fundamental concepts of CA as regards its transition function: (1) The basic (*uniform*) CA work with cells that share a single transition function. In this case the transition function of a cell can be considered as the transition function of the CA. (2) The *non-uniform* concept allows individual cells to determine their states according to different (*local*) transition functions. In both cases the behaviour of the CA arises from a cooperative update of all its cells during a sequence of development steps. The design of a suitable (efficient) transition function represents a key process aimed at achieving the desired behaviour of a CA. The cellular neighbourhood is defined uniformly for each cell, and its form primarily depends on the dimension of the CA. In the paper, one-dimensional (1D) and two-dimensional (2D) uniform CA will be considered whose behaviour is controlled by a deterministic transition function and the cellular neighbourhood is defined as follows. In the case of the 1D CA the neighbourhood of each cell is composed of the given cell and its immediate left and right neighbour (a 3-cell neighbourhood). Regarded as the cellular neighbourhood of the 2D CA will be the given cell and its immediate neighbouring cells in the north, south, east and west directions (a 5-cell neighbourhood). Cyclic boundary conditions will be applied due to the limitation of the CA size to a finite number of cells only for practical implementations. This means that the left-most and right-most cells in the 1D CA are considered as neighbours and similarly, in the case of the 2D CA, the opposite cells at the boundary of the cellular array in both dimensions are considered as neighbours. The CA will be assumed to work with more than two cell states and referred to as *multi-state cellular automata*.

A. Overview of Cellular Automata Literature

Many CA-based systems were successfully designed using analytical methods (for example, for the investigation into computational properties of the CA and construction of computing systems [10][11][12][13][14][15], development of replicating structures [16][17][18][19] or solving some specific mathematical operations and benchmarks in the cellular space [20][21][22][23][24]). However, 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, and due to the fact that the process of "programming" the cellular

This work was supported by the Czech Science Foundation via the project no. GA14-04197S Advanced Methods for Evolutionary Design of Complex Digital Circuits.

Dr. Michal Bidlo is with the Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence, Božetěchova 2, 61266 Brno, Czech Republic, e-mail: see http://www.fit.vutbr.cz/∼bidlom/index.php.en.

automata is not intuitive. In order to overcome this issue, the aim is to automate the process of designing (or identifying) the transition rules, using both deterministic algorithms and other heuristics or unconventional (non-deterministic) techniques, including evolutionary algorithms (EA).

Adamatzky proposed an algorithm for the identification of cellular automata rules from a given series of global transformations during a finite number of CA steps [25]. His approach was later improved, e.g. by combining it with Learning Classifier Systems [26] or advanced representation techniques like polynomial representation or decision trees [27]. Packard et al. were among the first who applied genetic algorithm (GA) [28] in order to adapt the transition rules [29][30]. Sapin et al. used an evolutionary approach to study the problem of discovering gliders in cellular array [31]. Sipper proposed a special technique, called Cellular Programming, for a parallel evolution of non-uniform CA, using a modified GA in each cell [32]. In recent years several works were published dealing with the design of cellular automata using, various evolutionary techniques. For example, Breukelaar and Bäck applied GA in order to evolve multi-dimensional uniform cellular automata to solve the density task and checker-board benchmarks [33]. They used ordinary table-based representation of the transition function and focused on tuning the GA settings, concluding that their system performed better than that in an earlier work published by Mitchell et al. [34]. Sapin investigated the evolutionary discovery of glider guns in cellular automata [35]. Elmenreich et al. proposed an original technique for calculating the transition function of CA, using neural networks (NN). The goal was to train the NN by means of Evolutionary Programming [36] in order to develop self-organising structures in the CA [37]. 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 [38]. Bandini et al. dealt with effects in cellular automata that may be observed by introducing asynchronous update schemes [39].

As regards research into multi-state CA in recent years, several studies using various bio-inspired techniques have been proposed. For example, in [40] a swarm intelligence algorithm, called Stochastic Diffusion Search, was applied as a tool to identify symmetry axes in patterns generated by cellular automata. The results provided a deeper insight into the emergent behaviour of CA and showed some interesting features (e.g. aesthetic qualities) of the generated patterns with potential applications in computer graphics. The authors say: "A two-dimensional multi-state cellular automaton with periodic boundary provides an endless environment for the growth of patterns and the observation of emergent complex behaviour over the time of evolution." This statement indicates the importance of (multi-state) CA research since the results may provide valuable information for the area of complex systems in general (especially the issue of emergent behaviour, which allows using simple rules in order to achieve a complex, cooperative global behaviour). Skaruz et al. published an approach to pattern and image reconstruction, using multi-state CA [41]. They applied the 3-state 2D CA with 9-cell neighborhood and showed that the genetic algorithm was able to discover satisfactory transition rules to reconstruct an image with up to 70% of damaged pixels. It is worth noting, however, that a certain margin of error can be tolerated in the case of image processing, which may reduce the complexity of searching for the rules. Baetens et al. studied the issue of stability and defect propagation in the development of 1D 3-state CA with 3-cell neighbourhood [42]. In particular, it was shown how the assessment of stability could be performed using non-directional Lyapunov exponents (based on a previous study proposed in [43]). The authors focused on a special class, called totalistic CA, in which the rules depend only on the total (average) of the cell states in a neighborhood.

The main focus of the paper (in comparison with the aforementioned studies) is on investigating both 1D and 2D CA working with at least six cell states. The transition function (to be discovered by evolutionary algorithm) is not limited to any specific class of CA (i.e. any arbitrary solution is acceptable that satisfies the conditions, i.e. the behaviour specified for the CA). The CA behaviour (evaluated in the fitness function of evolutionary algorithm) will be exactly given (for example as a minimal number of copies of a specific structure required to emerge within a given finite number of CA steps, an exact pattern to be developed from a given initial CA state, the result of a calculation encoded in a stable final CA state with respect to the initial state).

B. Motivation and Goals

Cellular systems demonstrated some advantageous features in solving various problems of both application-specific and generic nature. Although some automatic design techniques for cellular automata have been proposed and successfully validated on selected case studies and benchmarks, some limitations can be observed if a solution of a different kind or more general problem is needed. For example, Cellular Programming cannot effectively handle uniform CA that may be more interesting from the viewpoint of the physical implementation or control than the non-uniform model. The identification of the transition rules from a CA development sequence is not applicable if the behaviour of the CA is not known exactly. For the increasing number of cell states the solution space of the CA grows significantly, which makes the search for a suitable transition function difficult (the problem of scale). But it is not only for these reasons that the research into new (unconventional) design methods and representations is still important.

Considering the recent progress in physical theory and information technology, advanced models have been studied involving principles from quantum theory [44][45], nanoscale design [46][47], implementation on the molecular level [48] or combination of some of these principles for the FPGA design [49]. Another example could be the evolution of transport networks using CA models inspired by slime mould of a large cell called Physarum polycephalum [50]. This shows that cellular automata have become a model applicable in

current interdisciplinary areas for which new findings will be important even from the research on the elementary level.

The following scenarios can be identified regarding the design of transition rules for CA:

1) *Identification of the transition rules from a CA analysis.* This method assumes that a sequence of CA states is known (i.e. the CA development for a finite number of steps). The task is to identify the transition rules according to which the cells update their states.
2) *Design of the transition rules for given conditions in CA.* The exact CA behaviour is not known, the task is to design a transition function together with (a part of) the CA development that fulfils the given requirements (e.g. to achieve a specific state from a known initial state, to achieve a periodic or stable behaviour, etc.). This scenario will be considered in this paper.

The goal of this paper is to investigate the automatic evolutionary design of complex multi-state cellular automata, using an evolutionary algorithm combined with a special technique to encode the transition functions referred to as Conditionally Matching Rules. In order to demonstrate the abilities of this method, several different conditions required and evaluated in the CA will be considered. In particular, the problem of generic square calculation in 1D cellular automata will be presented as the first case study, whose results show that it is possible to substantially reduce the number of steps needed to calculate the square in comparison with the existing solution. The second (more complex) study will investigate the evolution of non-trivial replication processes in 2D CA taking into consideration three different loop-like structures. It will be shown that more efficient results can be generated using the Conditionally Matching Rules in comparison with the existing solutions. A novel replication scheme will be presented that exhibits a higher replication speed in comparison with the existing replicating loops. The utilisation of the Conditionally Matching Rules in this paper represents the first case of a successful evolutionary search for solutions to problems in multi-state cellular automata for which the conventional approaches have failed.

II. CA EVOLUTION USING CONDITIONAL RULES

For a successful CA design the representation (encoding) of the transition function represents a key issue. This section summarises the most important representations known from the literature and describes the principles and setup of Conditionally Matching Rules — an advanced representation proposed for the evolutionary design of multi-state CA — that will be considered for the experiments in this paper.

A. Overview of representations for cellular automata

A common (basic) approach to representing the transition rules is a table-based method where a rule specifies a new state for a given (single) combination of states in the cellular neighbourhood. For example, Packard [29][30], Sipper [32] or Breukelaar et al. [33] used this kind of encoding for the evolutionary design of CA. However, the table-based method is not suitable for multi-state CA because the complexity of designing such CA increases significantly with increasing number of states. Therefore, advanced representations have been investigated. Andre et al. utilised Genetic Programming (GP) [51], in which the transition function is encoded in a tree representing a *program* that calculates the updated cell states [52][53]. Other specific representations have been used for the identification of CA rules from a sequence of global states, e.g. polynomial representations [54] or decision trees [55]. The advanced representation techniques can provide some improvements over the basic approach, e.g. a reduction of the size of representation and computational complexity or increasing the modification (manipulation) efficiency. In particular, Bidlo et al. used a representation based on Linear Genetic Programming [56] to evolve CA and demonstrated an increased success rate and reduced computational effort over the table-based method for the replication and pattern development problem [57]. However, advanced experiments showed that this method was not suitable for more complex problems in which specific transition rules need to be applied. Therefore, the concept of conditionally matching rules was introduced as described in the following section.

B. Conditionally matching rules

Conditionally matching rules were introduced in [58] and their abilities demonstrated when solving a specific kind of the replication problem and a non-trivial pattern transformation problem in binary 2D CA, where the GP-based representation failed. Moreover, the potential of this method was demonstrated by further successful experiments regarding the evolution of multiplication in the uniform 2D cellular array [59]. The following paragraphs describe the setup of the conditionally matching rules that will be applied to the evolutionary design of multi-state celular automata.

A Conditionally Matching Rule (CMR) represents a generalised rule of a transition function for determining a new cell state (in view of the table rules). Whilst the basic transition rule specifies a new state for a specific combination of states in the cellular neighbourhood, a single CMR may cover more than one combination. A CMR is composed of two parts: a conditional part and a new state. The number of items (size) of the conditional part corresponds to the number of cells in the cellular neighbourhood. Let us define the condition item as an ordered pair of a condition and a state value. The condition is typically expressed as a function whose result can be interpreted either as true or false. The condition function evaluates the state value in the condition item with respect to the state of the appropriate cell in the cellular neighbourhood. In particular, each item of the conditional part is associated with a cell in the neighbourhood with respect to which the condition is evaluated. If the result of the evaluation is true, then the condition item is said to match with the cell state in the neighbourhood. In order to determine a new cell state according to a given CMR, all its condition items must match (in such a case the CMR is said to match). Figure 1 shows an example of a CMR defined for a 2D CA with 5-cell neighbourhood where ordinary relational operators $==, \neq, \leq, \geq$ are considered as the condition functions. Note

that these operators will be considered for all the experiments presented in the paper.

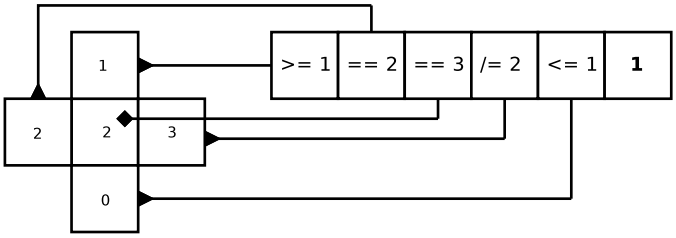

Fig. 1. Example of a conditionally matching rule working with 4 cell states and 5-cell neighbourhood. The right-most value of the CMR represents the new state (shown in **bold**).

A CMR-based transition function is considered as a finite sequence of conditionally matching rules. The algorithm of determining a new state is the following. The CMRs are evaluated sequentially, starting with the first CMR in the sequence. If a CMR compares the given cellular neighbourhood, then this CMR is used to determine the new state of the cell to be updated. Note that, because of the sequential evaluation of the CMRs, it is always the first mathing CMR in the sequence. If none of the CMRs matches, then the cell keeps its current state. This approach ensures that the process of calculating the new state is deterministic (on the assumption that the condition functions are deterministic too). The aspect of determinism is important in order to preserve the traditional concept of CA, and with respect to the applications investigated in the paper. In the case of the deterministic CA, the process of development ensures that for a given configuration the CA produces a specific behaviour (result of the development). Specifically, the result of the square calculation has to be the same for a given value, and for the replication of a loop it is expected the the copy is the same as the original structure. Moreover, a deterministic (CMR-based) transition function can be transformed into a corresponding table-based representation, which allows us to use a conventional form of the transition rules implemented in many existing CA simulators. Another advantageous feature of the CMRs is the ability to specify conventional (table-based) rules if needed. In order to do that, the relation $==$ with given state values specified in each item of the conditional part of a CMR allows specifying a transition rule for a given (single) combination of states in the cellular neighbourhood.

An example of a 1D CA step according to a sample CMR-based transition function is shown in Figure 2a,b. Note that the cell states are updated synchronously, i.e. the cells evaluate the transition function in parallel with respect to the actual CA state (2 1 2 3). Cell $c1$ will not change its state because no CMR matches the cellular neighbourhood 021. For cell $c2$ only CMR $r2$ matches the neighbourhood 212, therefore, the new state of this cell will be 2. Cell $c3$ will get its new state according to CMR $r3$, which matches the neighbourhood 123. Finally, rules $r2$ and $r4$ match the neighbourhood 230 of cell $c4$, therefore CMR $r2$ will change the state of $c4$ to 2.

Considering the aforementioned example with a 4-state CA and 3-cell neighbourhood, the complete table of the transition function consists of $4^3 = 64$ rules (each rule for a single

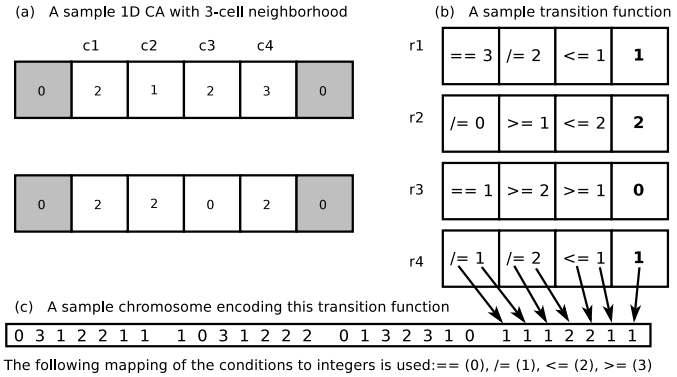

Fig. 2. (a) Example of a multi-state 1D CA working with 4 cell states, 3-cell neighbourhood and zero boundary conditions. A single step performed according to a CMR-based transition function from part (b) is illustrated. The corresponding chromosome of this transition function is shown in part (c) together with encoding the condition functions as integer values.

specific combination of states in the cellular neighbourhood). This means there are in total 4^{64} possible transition functions (possibilities of what next state can be specified for each combination). However, the CA often involves only a subset of rules — combinations of neighbourhood states — for which a different state is specified with respect to the current state of the cell to be updated (these rules need to be explicitly specified in the transition function). For the purposes of the evolutionary design it would be possible to shorten the chromosomes of the evolutionary algorithm (and thus to reduce the search space) if the subset of explicit rules were known. Unfortunately, it is difficult to effectively predict these rules for a given application if the CA development is not known exactly. In order to reduce the size of the chromosomes needed to encode the transition function, the CMR approach was introduced. If the example from Figure 2 is considered (4 cell states, 3-cell neighbourhood and 4 condition functions), each CMR can be encoded as a 7-tuple of integers. For a transition function consisting of 4 CMRs, a candidate solution can be represented by $4 \times 7 = 28$ integers, each of which may acquire 4 different values. Therefore, the search space can be reduced substantially to 4^{28} possible solutions. Our hypothesis is that the CMRs allow exploring more effectively the search space and discovering solutions to some problems that were not achieved using the conventional representation.

C. Design of CMR-based CA using evolutionary algorithm

In the paper the search for a suitable sequence of CMRs satisfying a given CA behaviour is performed using a population-based EA. The EA has been chosen after a long-term experience with the CA design, and its setup is based on that proposed in [58]. Our preliminary experiments with this EA showed that its simple concept and computationally efficient mutation operator represent the bigest advantages (potentially suitable for accelerated hardware implementations in the future). The small population size allows performing a satisfactory amount of iterations (hundreds of thousands to several millions, depending on the specific experiment) in order to evolve the candidate solutions whose evaluation takes

a long time with respect to the computational effor of the EA (this holds for the evaluation of CA performing several tens of development steps). Note that a detailed investigation and optimisation of the EA operation are not the subject of this paper.

A population of 8 chromosomes is considered, each of which represents a candidate transition function encoded as a finite sequence of conditionally matching rules. Each CMR is encoded as a finite sequence of integers representing the conditional parts (i.e. the condition functions and state values) and the next state as illustrated in Figure 2c. Note that the chromosome is represented as a linear array of integers directly coding the condition functions and state values for each CMR. The chromosomes for the initial population are generated randomly at the beginning of the evolution. The evaluation of each chromosome is performed using a CA whose development (controlled by the transition function encoded in the chromosome) from a given initial state is observed with respect to the required behaviour for a finite number of steps. The objective function that calculates the fitness values of the chromosomes is specific for different case studies and will be described for each experiment in Sections III and IV.

In each generation of the EA a new population is created according to the following algorithm: Four chromosomes are selected randomly, the best one of which is considered as a parent (i.e. fitter chromosomes are preferred to generate offspring) – it is a case of the tournament selection with base 4. An offspring is created by mutating 0–2 randomly selected integers in the parent, which is performed via replacing the selected integers by new valid randomly generated values. The number of integers to be mutated is also selected randomly; if 0 integers are chosen, then the offspring is identical to the parent. The chromosome selection and mutation continue until the new population of the same size is filled by the offspring. If a solution is found (after evaluating the chromosomes in the new population) that satisfies the given CA behaviour, then the evolution is considered as successful. If no solution is found within a maximal number of generations (which is specific to various experiments), then the evolution is terminated.

III. Experiments with square calculations

The first case study considers the evolutionary design of 1D cellular automata whose development can be interpreted as calculating the square of a number. The basic idea to perform this operation in the CA (together with choosing the appropriate number of cell states, representation of input values and way of evaluating candidate solutions) has been inspired by Wolfram's work [23], section Computations in Cellular Automata, page 638. The computation of x^2 is interpreted as a CA development that comes (after a finite number of steps) into a stable state in which the result for the given x is encoded.

In the proposed experiments a 1D CA consisting of 100 cells and working with 8 cell states will be considered. The value of x will be encoded in the initial CA state as a continuous sequence of cells in state 1, whose length corresponds to x, the other cells possess state 0. For example, a 10-cell CA encoding $x = 3$ can appear as 0001110000. The result is assumed to be a stable CA state in which a continuous sequence of cells of a single state different from state 0 can be detected whose number equals x^2, the other cells are required to be in state 0. The goal is to discover cellular automata that are able to calculate the square of an arbitrary number $x > 1$.

Two scenarios of the fitness evaluation are investigated: in the first case the values of x from 2 to 5 are evaluated during the evolution whilst the second setup considers x from 2 to 6. This approach is motivated by the assumption that if the CA is required to work for more values of x during the evolution, then it will be harder for the EA to design such a CA (i.e. the success rate will be lower) but, on the other hand, more general solutions could be obtained (i.e. such CA that, using the evolved transition function, are able to correctly calculate the square for any higher number not considered in the fitness evaluation). The result of the x^2 calculation in the CA is evaluated after the 99th and 100th steps in order to determine whether the resulting state is stable. Considering the aforementioned setup the fitness of a fully working solution for x from 2 to 5 is given by $F_{max2-5} = 4 * 2 * 100 = 800$ (4 different values of x are considered, the result of each is evaluated for the last 2 steps in a CA consisting of 100 cells), the second scenario considers the maximal fitness $F_{max2-6} = 5 * 2 * 100 = 1000$. The maximal limit of generations for these experiments was set to 200,000. For each scenario 100 independent evolutionary runs were executed. The success rate and the average number of generations needed to find a working solution were measured. In order to identify general solutions for the square calculation, the resulting CA were validated for the values of x up to 100, using the transition functions obtained from the successful evolutionary runs. For the purposes of the paper the solution that passed this test is considered as general.

Table I summarizes the statistics of the evolution for this set of experiments. As evident, the success rate is substantially lower if more values of x (i.e. from 2 to 6) are evaluated. In this case approximately half the evolutionary runs finished successfully in comparison with the scenario with x from 2 to 5. However, more general solutions were obtained as shown in the last row of Table I, which confirms the assumption stated in the previous paragraph. Although the general square calculation definitely cannot be regarded as a trivial task for uniform CA, the success rate and the resulting number of generations indicate that the proposed CMR encoding of the transition functions represents an efficient technique of solving this problem for reasonable values of x considered during the fitness evaluation.

Table II shows the number of steps of the best evolved solutions needed to obtain the result of x^2 in the CA. The results obtained are compared to Wolfram's solution published in [23]. The number of CA steps determines the efficiency of the square calculation (i.e. the fewer steps, the faster the calculation). From this point of view the best result obtained in this paper (denoted as Solution #3 in Table II, whose CA is controlled by 432 rules) is more than twice faster in comparison with Wolfram's CA (see the last row of Table II). For example, Solution #3 calculates 5^2 in 32 steps whilst

TABLE I
STATISTICS OF THE EVOLUTIONARY EXPERIMENTS DEALING WITH THE CALCULATION OF x^2 IN 1D CELLULAR AUTOMATA. THIS SET OF EXPERIMENTS CONSIDERED CA WORKING WITH 8 CELL STATES.

x eval. →	from 2 to 5			from 2 to 6		
#CMRs	Succ. rate	Avg. #gen.	Std. dev.	Suc. rate	Avg. #gen.	Std. dev.
20	57	60709	54704	26	72648	48215
30	64	60293	50602	28	79299	65138
40	59	67539	55114	26	85865	57281
50	57	61342	52561	32	80526	51720
#general		4			6	

Wolfram's CA needs 78 steps. An example calculating 5^2 using our best solution is illustrated in Figure 3.

TABLE II
ANALYSIS OF SELECTED RESULTS FOR THE SQUARE CALCULATION. THE TABLE SHOWS THE NUMBER OF STEPS OF THE CA NEEDED TO OBTAIN THE RESULT OF x^2 FOR x FROM 2 TO 9 USING THE BEST OBTAINED RESULTS AND WOLFRAM'S SOLUTION [23].

x	2	3	4	5	6	7	8	9
Solution #1, 400 transition rules								
#steps	6	12	24	40	60	84	112	144
Solution #2, 408 transition rules								
#steps	5	12	20	34	48	68	88	114
Solution #3, 432 transition rules								
#steps	4	10	20	32	48	66	88	112
Wolfram's solution								
#steps	12	28	50	78	– not available –			

An analysis of the results has shown that all the general solutions exhibit a regular pattern generated by the CA during its development (which is actually inevitable in order to correctly calculate the result for various x using the same transition function of the CA). However, it can be observed that various approaches can be discovered that differ in both the efficiency and the complexity of the CA. Moreover, one of the key features is the way of encoding the input value and the result. Whilst Wolfram used a method of representing these values by two different (non-zero) states in the CA, the encoding proposed in this paper considers only a single state and allows the result to be represented by a different state than the input value is encoded by (which probably enabled a more efficient square calculation in comparison with Wolfram's solution). These observations indicate that research into advanced techniques of input and output representation could provide further interesting solutions to this problem.

IV. EXPERIMENTS WITH REPLICATING LOOPS

The second case study considers a class of 2D cellular automata that are able to replicate a given structure. Replicating loops represent one of the typical benchmarks that have been studied in relation with cellular automata. There are various variants of loops that differ in shape, size, complexity or replication speed. In most cases cellular automata that perform replication of the currently known loops were designed using

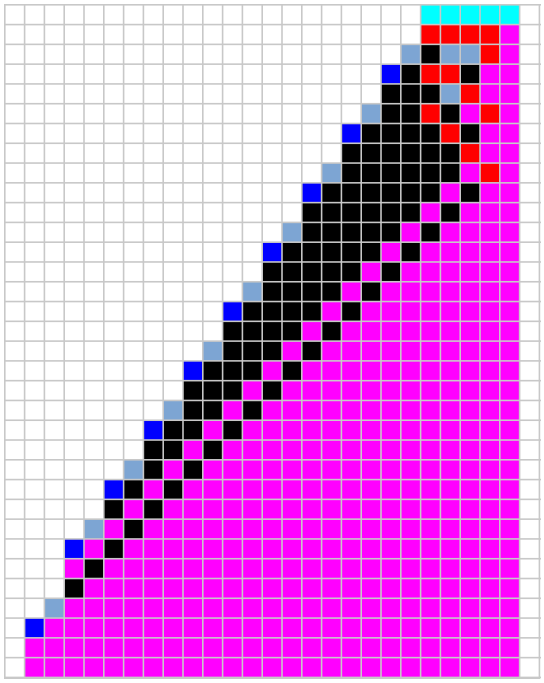

Fig. 3. Calculation of x^2 in one of the best CA discovered by evolution ($x = 5$ in this example). This CA works with 432 transition rules, 32 steps are needed to obtain the result (note that Wolfram's solution [23] needs 78 steps).

analytical approaches (i.e. the transition rules were specified manually after an in-depth understanding of the loop structure and the transformation process leading to the creation of its copy). In this section we demonstrate that it is possible to automatically design transition functions for various replicating loops. In particular, it will be shown that the EA can discover a completely new replication scenario whose replication speed is several times higher in comparison with some currently known loops. Note that for the purposes of this paper the *replication speed* will be considered as the number of copies of the loop that can be created depending on the number of CA steps executed. The reason for the utilisation of this metric is that it allows quantifying an amount of objects (e.g. loop structures) the CA is able to create in a given number of steps and comparing its behaviour with other known replicating structures. In general, the speed of the CA development could, for example, express the area of the cellular array the CA is able to cover (or to process), taking into consideration some constraints and conditions of a specific application.

Experiments with the replicating loops were conducted using the EA described in Section II-C, for which the maximal number of generations was set to 3 million. The evolution time of a single run is approximately 12 hours when using the Anselm cluster, which is a part of the Czech National Supercomputing Center[1].

Several sets of experiments were conducted with various replicating loops. Figure 4a,b,c shows the loops that were considered in our experiments. For the purposes of this paper

[1]http://www.it4i.cz/?lang=en
https://docs.it4i.cz/anselm-cluster-documentation/hardware-overview

the loops will be denominated by symbolic names as (a) round loop, (b) rectangular loop and (c) envelope loop. Note that the loop shapes as well as the numbers of cell states proposed for specific expreriments were chosen on the basis of the existing replicating loops. In addition, the objective is to determine the ability of the EA and CMRs to tackle various ways of the fitness evaluation and specification of the target CA behaviour.

The round loop and rectangular loop share a common principle of the fitness evaluation based on the known shape and size of the loop. Figure 4d illustrates the concept of the fitness evaluation containing a rectangular loop as an example (note that the CA size was chosen with respect to a reasonable space for creating the replicas in various directions). As shown by the textured cells, the replicas are expected to be in a regular grid with respect to the initial loop. This scenario was applied on the basis of some existing replicating loops that work in a similar way. The replicas are required to be separated from each other by a boundary consisting of cells in state 0. Thus in the case of the rectangular loop in Figure 4d the complete rectangle of a replica to be evaluated consists of 6x6 cells as marked by a thick rectangle. In order to evaluate a candidate solution, the CA development is analysed for 30 steps. After each step of the CA the following calculation is performed. The *partial fitness* is calculated separately for each replica in the grid as the number of cells in correct state with respect to the loop rectangle. Then the *step fitness* is defined as a sum of all partial fitness values for a given step of the CA as follows. If the partial fitness equals the number of cells in a given rectangle (i.e. a perfect replica is detected), then the step fitness is increased by the double of this number (as a bonus for finding a replica), otherwise the step fitness is increased by the original partial fitness value. The final fitness is determined as the maximal step fitness out of all the CA steps considered for the evaluation. If at least 4 perfect replicas are detected in a state within the 30-step development, the evolution is finished successfully.

In order to validate the results, a software simulator developed by the author of this paper is applied. A larger CA is executed for more than 30 steps using the evolved transition functions, and a visual inspection is performed in order to identify *general replicators*. For the purposes of this paper, each solution with the ability to persistently produce replicas according to the specification in the fitness function is classified as a general replicator.

Statistical results of the replication experiments are summarised in Table III. For each type of the replicating loop the success rate and computational effort (expressed as the average number of generations needed to find a working solution) were evaluated depending on various selected CMR counts. The number of general solutions is determined out of all successful experiments performed for a given loop. Setups with 6 and 8 cell states were considered for the rectangular and the round loop. The results show that 30 or 40 CMRs are in most cases adequate to find a working solution. For 20 CMRs the success rate is usually very low, probably due to a lack of resources (rules available for the CA) that can be expressed by the CMRs. On the other hand, 50 CMRs induced a huge search space that is very time consuming for the EA to explore

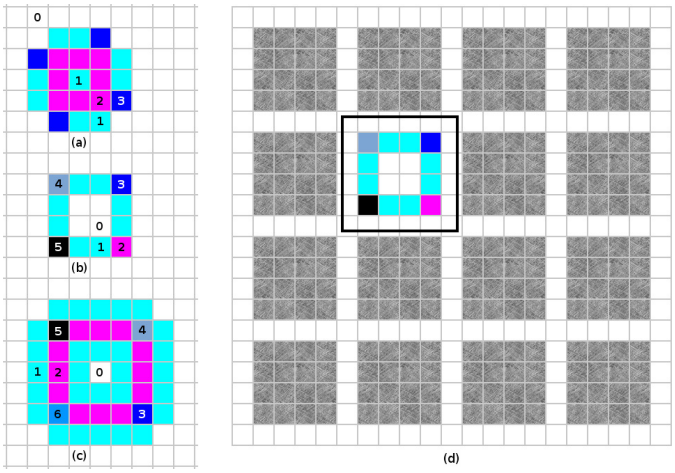

Fig. 4. Loop structures considered for the replication experiments (the corresponding state values of cells are also shown): (a) round loop, (b) rectangular loop, (c) envelope loop, (d) illustration of a fitness calculation scheme used for the round and the rectangular loop. The textured cells are evaluated with respect to the required replica structure. A boundary of white cells in state 0 separates the replicas. The initial loop is marked by a thick rectangle.

(i.e. the success rate is low with a given generation limit). Although the success rate achieved in general is under 20%, these experiments showed for the first time the possibility of automatically designing complex multi-state CA using the CMR encoding. Note that we were not able to successfully design any of the CA that use conventional representations of the transition functions.

The results in Table III also show that most of the experiments require on average more than a million generations in order to find a working solution. This means that the replication does not represent a trivial task for the EA so that a significant part of the search space needs to be explored. Although the population works with 8 individuals only (this may justify the need for a higher number of generations), the time needed to evaluate the candidate solutions is considerable, which is not feasible for larger populations. The advantage of such a small population is also supported by the fact that in the proposed EA a single parent is chosen from 4 individuals (i.e. from half the population) and this parent produces offspring for the next generation. Experiments showed that a larger population did not improve the success rate because most of the time was spent on the fitness calculation, with no significant increase in the exploration of promising parts of the solution space. Several parameters can be tuned in the proposed EA: population size, tournament selection base, and the number of genes to be mutated. The optimal setting is usually problem dependent, which requires tuning the EA separately for different case studies. However, the tuning of the EA was not a goal of this study – the objective was to evaluate the proposed EA with CMRs using several different tasks in CA. In the case of successful experiments the tuning of the EA and examination of its features for various problems can be performed, which represents a topic for the future research.

Most of the general results obtained in this paper replicate the given loop in a single direction only. However, in some

TABLE III
STATISTICS OF EXPERIMENTS DEALING WITH EVOLUTION OF THE REPLICATING STRUCTURES FROM FIGURE 4. THE RESULTS ARE CALCULATED OUT OF 100 INDEPENDENT EVOLUTIONARY RUNS.

Loop →	**Rectangular 6 states**			**Rectangular 8 states**			**Round 6 states**			**Round 8 states**			**Envelope 8 states**		
#CMRs	**Succ. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**	**Suc. rate**	**Avg. #gen.**	**Std. dev.**
20	6	1320293	485774	7	1750754	714081	1	546936	-	3	1113809	983806	2	2655646	208081
30	5	1719899	871523	13	1404925	728908	12	1152757	674883	12	993487	631610	5	1626788	595484
40	12	1022548	815782	14	1056619	783897	12	804928	544980	19	867039	579131	11	1603042	706082
50	6	686951	410334	12	1195469	1110617	2	1402899	865896	10	841832	698793	9	976695	672486
#general		5			3			11			20			20	

cases the EA discovered a novel approach that can be used to effectively replicate the given structure from many (previously created) instances simultaneously. The best results obtained for each loop are described in the following subsections.

A. Results for the round loop

Although the round loop itself consists of only 4 different states (including the "empty" state 0 – see Figure 4a), the target CA works with 8 cell states. This setup was chosen in order to increase the probability of discovering various replication scenarios, which was the main goal of these experiments. For this experiment, a minimum of 4 replicas were required to emerge (potentially including the initial loop) within a maximal number of 30 development steps. Note that during the evolution the initial loop was initialised in an inner part of the CA (as shown in Figure 4d) in order to allow the EA to discover replication in any direction. This setup is motivated by some of the existing replicating loops that work in a similar way (i.e. the replicas can "grow" in more directions simultaneously).

The transition function of one of the best evolved solutions for the round loop is shown in Figure 5, and the CA development performing the replication process is illustrated in Figure 6. This CA represents the best result of this paper and will be referred to as Bidlo's loop. As evident from Figure 6, the first replica is created after the 12th step on the right (east) from the initial loop. However, some cells are emerging both in the east and south *before* the first replica is finished. These cells actually represent a basis for the next replicas being created concurrently during the subsequent development. From this process it can be seen that the evolution discovered a new replication scheme – let us call it a *diagonal replication* as it creates successive copies of the loop diagonally between the east and south directions. The following stages can be identified with respect to the overall CA development: (1) The first replica emerges after the 12th step on the east side. (2) This replica produces the first diagonal level after the next 8 steps. (3) Every next diagonal level is finished after 6 steps. Note that the development pattern of the CA is regular since stage 3, i.e. the number of replicas can be predicted and an analysis of the global CA behaviour can be performed.

For comparison purposes, Byl's loop was chosen [17] because of its closest similarity to the round loop with respect to the shape, size and complexity (see the illustration of

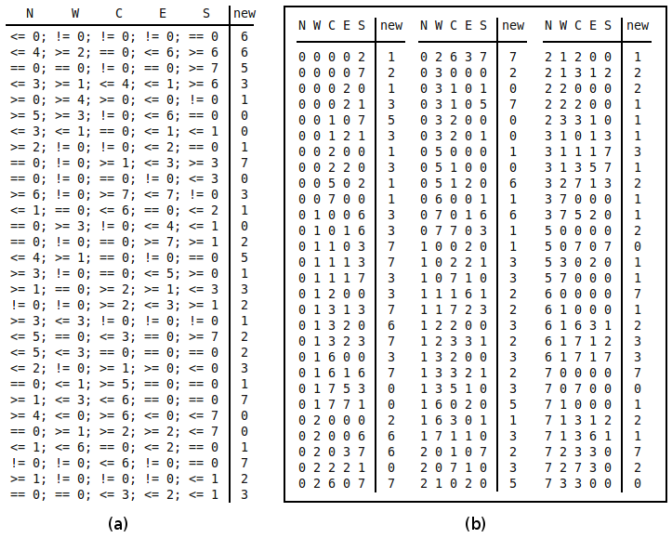

Fig. 5. Transition function for the replication of the round loop from Figure 6: (a) the CMR representation discovered by the EA, (b) the corresponding table-based representation.

the loops in Figure 7). Both these loops were analysed in detail, using our cellular automata simulator, with the following results. The transition function of Byl's loop consists of 238 rules, our solution for the round loop works with 84 rules. The CA simulator can provide integer sequences containing the numbers of replicas at the respective time (i.e. after executing the CA for the appropriate number of steps). For example, the CA with Bidlo's loop (discovered in this paper) can produce $3, 6, 10, 15, 21, 28, 36, 45, \ldots$ replicas in $20, 26, 32, 38, 44, 50, 56, 62, \ldots$ steps, respectively. In order to mathematically express the number of Bidlo's loop depending on the number of CA steps N, a manual analysis of these sequences was performed; the result is described by equation (1):

$$C_{Bidlo} = \frac{N^2 - 10N + 16}{72}; N \geq 20 \wedge (N-20)\ \%\ 6 = 0 \quad (1)$$

A similar analysis was performed to express the number of Byl's loop as described by equation (2):

$$C_{Byl} = \frac{2N^2}{625} - \frac{8N}{25} + 15; N \geq 100 \wedge N\ \%\ 25 = 0 \quad (2)$$

Note that the sign $\%$ in the formulas denotes the modulo division. Additional conditions for N are specified in order

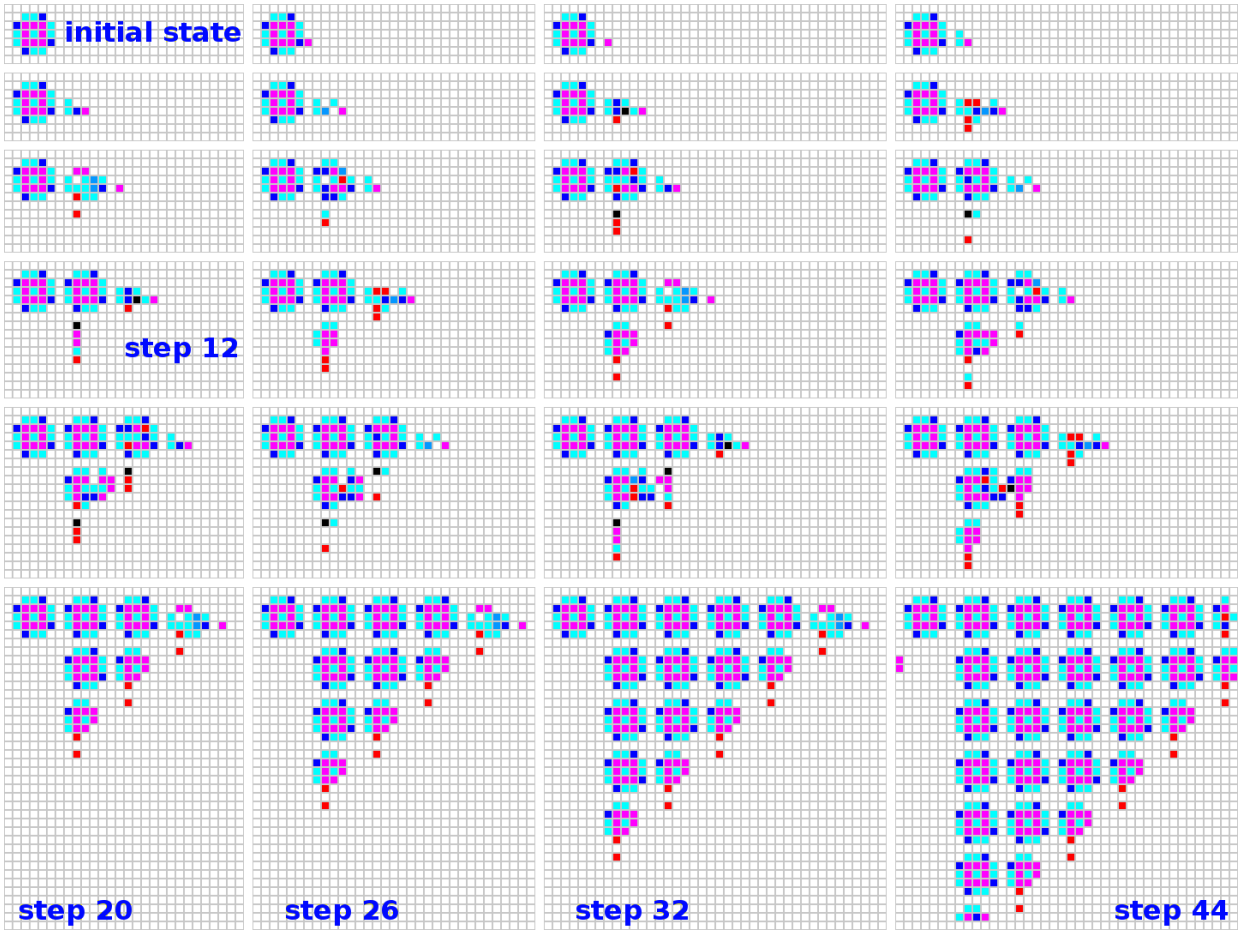

Fig. 6. Replication of the round loop in a CA developing according to the transition function from Figure 5. The sequence of CA states reads from left to right, top to bottom.

to express only the numbers of steps after which a group of replicas has just been completed (for other values of N the result is not a whole number). The correctness of the equations was confirmed using our CA simulator. Alternatively, suitable mathematical software can be involved. For example, WolframAlpha[2] represents a straightforward tool for such an analysis. By entering a part of the sequence containing the number of replicas for Byl's loop $(15, 25, 39, 57, 79, 105, 135, 169)$, WolframAlpha provides a mathematical expression for this sequence as shown by equation (3):

$$C_{Byl-WA} = 2x^2 + 4x + 9; x = 1, 2, 3, 4, 5, 6 \ldots \quad (3)$$

Note that in this case x represents an independent variable, not the number of the CA steps. Similarly, the number of replicas of Bidlo's loop can be expressed independently of N as shown by equation (4), which was derived by WolframAlpha:

$$C_{Bidlo-WA} = \frac{1}{2}(x+1)(x+2); x = 1, 2, 3, 4, 5, 6 \ldots \quad (4)$$

Again, our software simulator was applied that confirmed the correctness of the equations provided by WolframAlpha with respect to the output from the corresponding CA.

In the case of Byl's loop a group of replicas is completed after every 25 steps, for the round loop a whole diagonal

[2]http://www.wolframalpha.com/

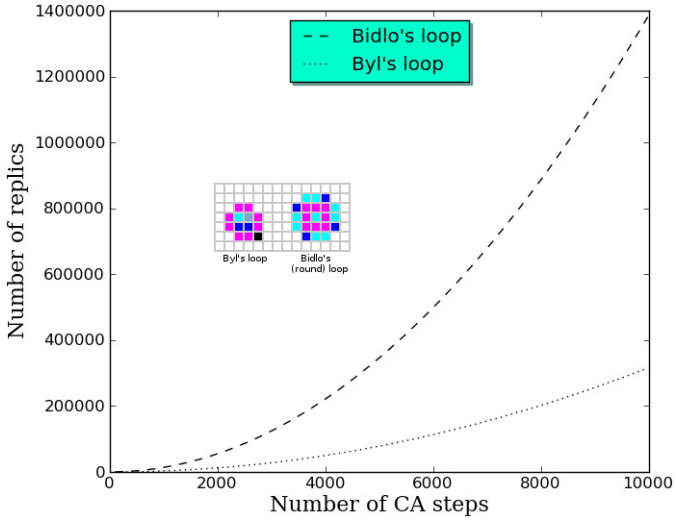

Fig. 7. Comparison of the number of replicas based on the number of CA steps for Byl's loop [17] and Bidlo's (round) loop.

of replicas is finished after every 6 steps. Although Byl's loop replicates into four directions, its replication speed is significantly lower in comparison with the round loop, which replicates in two dimensions only (see Figure 7). This feature

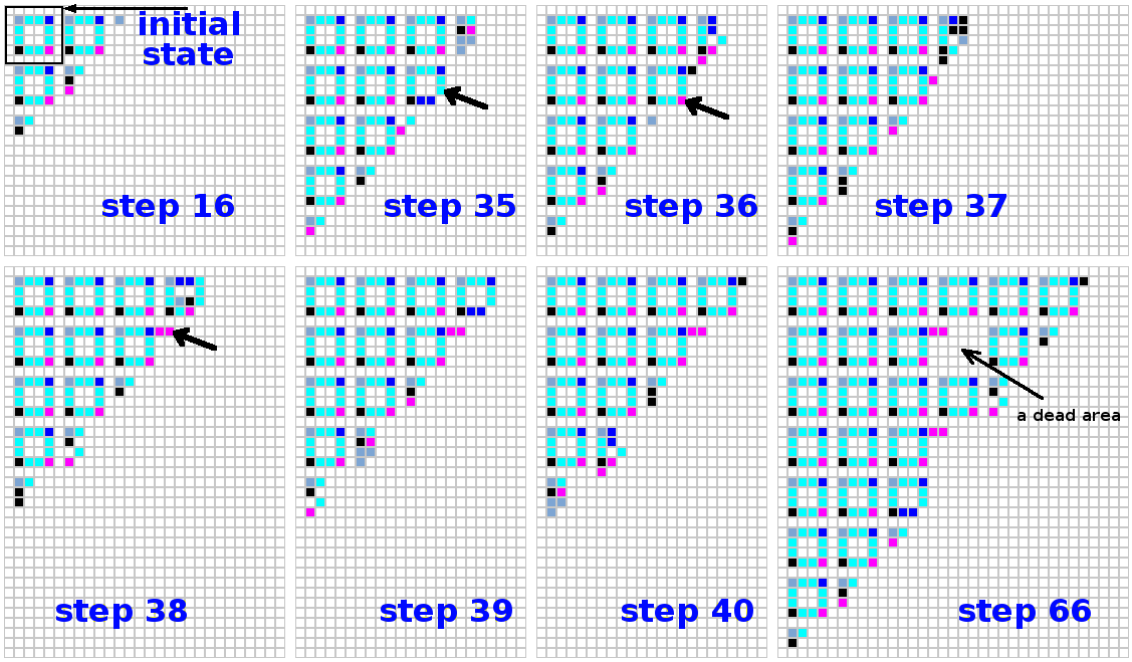

Fig. 8. Example of development of the rectangular loop discovered by the evolution. The CA works with 8 states, the sequence of steps reads from left to right, top to bottom. The initial state is marked by a thick rectangle in the top-left part of this figure.

follows from the significantly lower factor of the element N^2 in Equation (2) compared to Equation (1) for the round loop. This means that the solution obtained in this paper is significantly more efficient (from the point of view of both the replication speed and the complexity of the transition function) compared to Byl's loop. Considering the replication period (i.e. the number of steps after which a new group of replicas is completed), which determines the replication speed, our solution even overcomes (with its replication period of 6 steps) all the best known replicating loops. For example, in addition to Byl's loop [17] with a replication period of 25 steps, Chou-Reggia's loop [18] has a 15-step replication period, Langton's loop [2] replicates in every 151 steps or Perrier's loop [60] exhibits a replication period of 235 steps. Therefore, Bidlo's (round) loop can be considered as the fastest replicating loop known so far and represents the main result of this paper.

B. Results for the rectangular loop

In order to show that the CMR encoding of the transition function is not limited to a specific loop only, other structures were also considered. The rectangular loop from Figure 4b uses the same principle of the fitness evaluation as the round loop does. Several perfect solutions were obtained in which the aforementioned concept of diagonal replication can also be observed. One of those solutions is depicted in Figure 8.

In this case the development starts by a stage (1) that takes 16 steps during which two different replication processes are performed in both the east and south direction (the result of this stage is shown in the top-left part of Fig. 8). The next groups of replicas are developed during stage (2) after every 13 steps. However, it can be observed that the replication into the two directions is not symmetric – see Figure 8, step 35 (the first row of replicas contains 3 complete loops whilst the south direction contains 4 loops in the first column). This irregularity causes that some loops are not able to finish their replication, which leads to some "dead areas", which emerge during the CA development. Since step 35 it is evident that the loop marked by the black arrow starts its east replication in step 36 but the "arm" constructed in step 38 (consisting of

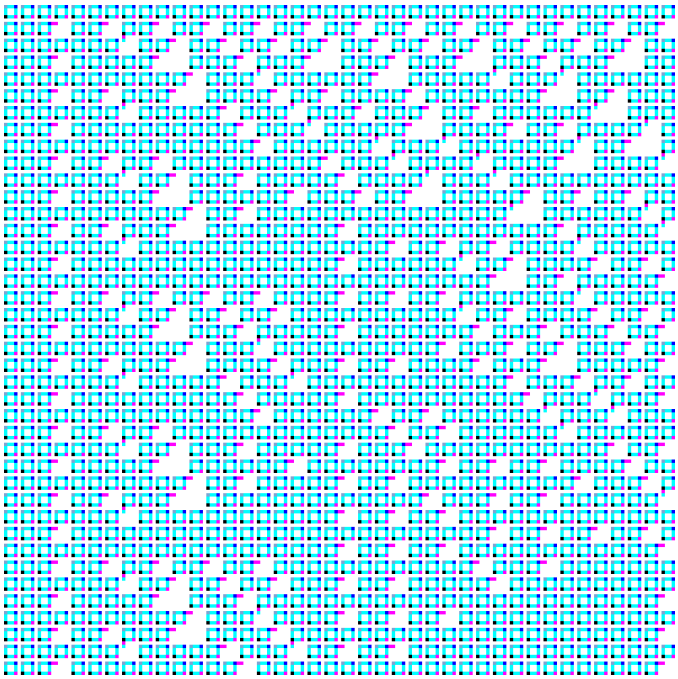

Fig. 9. A sample state of a 200x200-cell CA performing a replication of the rectangular loop. The state represents a continuation of the development from Figure 8 after 871 steps and shows a chaotic arrangement of dead areas caused by an asymmetric diagonal replication.

two cells in state 3) no longer develops during the next steps, producing a dead area that can be clearly visible after step 66. In order to identify the consequences of this anomaly a 200x200-cell CA was considered with the initial loop in its top-left corner. The development takes 871 steps until the CA reaches its stable state with the whole cellular space filled by replicas and dead areas (see Figure 9), whose arrangement seems rather chaotic. Moreover, the development of this loop was observed using 400x400- and 800x800-cell CA with no systematic arrangement of the loops and dead areas that would allow predicting reliably the global behaviour of such a CA.

Although the chaotic behaviour is quite common for CA in class 3 of Wolfram's classification [23], it has been rarely observed in the case of replicating structures. However, the issue of prediction and potential applications of such behaviour still represent open problems for the future research. Note that some other solutions were obtained that are able to replicate the rectangular loop without malformations, using the diagonal replication scheme.

C. Results for the envelope loop

The last experiment from the area of replicating structures demonstrates another approach to the fitness evaluation and considers a more complex replicating loop denominated as envelope loop – see Figure 4c. In this case the CA works with 8 different cell states, the initial state consists of a single instance of the loop to be replicated (see the top-left state of Figure 10 denoted as "initial state"). Contrary to the previous experiments, this replication task is evaluated using a fixed (reference) pattern as a complete CA state containing a copy of the original loop in a specific position (see the bottom-right state of Figure 10 denoted as "step 20"). The reason for this experiment is to determine the ability of the EA to evolve CMRs for the transformation into an exact copy arrangement and to identify whether general replicators are possible in this case (i.e. the ability of the CA evolved to repeatedly generate replicas according to the original specification if the development continues). Since the fitness evaluation is different, compared to the previous loops, the success of this experiment also indicates that the CMR approach is robust.

The step fitness function is calculated after each step of the CA as the number of cells in correct states with respect to the reference pattern. The final fitness is defined as the maximum out of all the step fitness values. From a more general point of view the evaluation of the candidate solutions may be considered as a pattern transformation problem transforming the initial state into another target state containing the replica. The results are finally verified in order to determine whether the transition function is able to generate more replicas if the CA development continues. Figure 10 shows an example of a successful solution performing a complete development of a single copy of the envelope loop. This solution is able to generate further replicas from the most recent one whereas each new replica is "shifted" by two cells down from its predecessor (as originally specified by the reference pattern during the evolution). The new replica is finished after 20 development steps and this solution represents the fastest replicator out of the results obtained for the envelope loop. The transition function consists of 102 table-based rules (transformed from the evolved CMR encoding). Note that various perfect results were obtained, for example a CA working with 88 rules (the most compact transition function so far obtained for this loop) generating replicas in 25 steps, another solution uses 96 rules and the replication takes 30 steps.

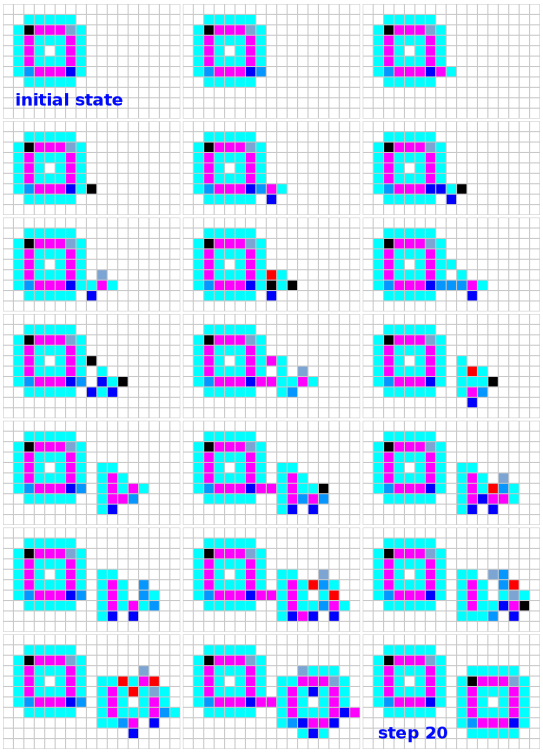

Fig. 10. A CA development performing the replication of the envelope loop. The sequence of CA states reads from left to right, top to bottom.

D. Discussion

The EA provided working solutions that satisfy the given requirements in all the experiments, which indicates that the CMR approach is robust. Since the fitness function represents one of the key aspects of a successful evolutionary system (together with the representation of candidate solutions), it may enable the CMR concept to be successfully applied in advanced CA models in various areas (e.g. non-uniform, asynchronous or probabilistic CA).

An observation of the evolved CA development showed that different replication techniques could be obtained even for a specific loop. For example, in addition to the results proposed in Section IV-A, a CA working with 6 cell states was discovered that needed the same number of steps (12) to replicate the initial loop but its transition function contained 99 rules (the solution from Section Section IV-A works with 8 cell states and 84 rules, Byl's loop replicates according to 238 rules). In the case of both the round loop and the rectangular loop the replication process designed by the EA exhibits a pipelining principle (i.e. the replication of the next group of loops is in progress before the previous group has been finished). Although the replication process works for a specific

direction only (e.g. it does not allow an easy modification or adaptation of the transition rules by "rotating" the cellular neighbourhood known from Byl's loop [17]), it has been shown that the proposed loops replicate with a higher speed against the existing loops. Another important aspect discovered by the EA, which probably contributes to the replication speed, is the diagonal replication. Moreover, it seems that the diagonal replication represents a technique that enabled reducing the number of transition rules needed to perform the replication. In particular, the proposed Bidlo's loop can create 528 instances in 200 CA steps compared to 79 instances of Byl's loop in the same time (i.e. in this case the proposed loop produced more than 6 times more copies than Byl's loop).

Although the form or arrangement of the proposed loops was inspired by the existing (self-replicating) loops, the results obtained do not exhibit the concept of self-replication in which the information needed to create the replica is encoded in the loop body. In fact, no exact solution to any existing self-replicating loop has yet been rediscovered using the proposed approach. The information encoded in the loop, which specifies the self-replication features, determines the CA development, which is specific to the given loop (e.g. Langton's loop encodes a sequence of states that determine how to create the replication "arm", when to turn the arm in order to create the corners of the loop or how to "close" the final replica). The transition function must interpret this information in order to control the CA development accordingly. The problem is that if such a transition function is specific to a given loop (i.e. there are no or only very few transition functions in the solution space that would perform self-replication of the given loop), then the EA may not be able to find the solution in a reasonable time.

However, the proposed approach allowed discovering transformations for creating replicas that are completely new or even exhibit some phenomena that would probably not be intended or acceptable during a manual CA design. In order to enable the EA to design innovations, it is important for the evaluation of the candidate solutions not to be very strict. In particular, the diagonal replication scheme, pipelining principle or malformation of the loops during the replication process represent phenomena designed by the EA that were not explicitly specified (required) by the designer.

V. Conclusions

In this paper some selected applications of conditionally matching rules were presented for a routine evolutionary design of complex multi-state uniform cellular automata. The proposed EA in combination with the CMR encoding was able to find working solutions to problems for which the conventional representation techniques failed. In particular, a novel replication scheme in 2D CA was discovered in this paper that allows a faster development of the copies of the given structure in comparison with the known approaches. Some techniques were evolved to calculate the square of integer numbers in 1D CA that require a considerably lower number of steps compared to the existing solution. The results obtained represent the first case of a successful automatic design of multi-state CA for this kind of problems.

In general, the proposed approach showed the ability to find working solutions to various kinds of problems (the square calculation in 1D CA and replication in 2D CA, where the result can be specified either as a minimal number of replicas or by a fixed pattern). This indicates that other classes of problems could be successfully solved in the future. For example, the CMRs might allow optimising the pattern generation in the area of computer graphics. Other potential applications may include the design of CA for random number generation or the optimisation of test pattern generators for digital circuits.

As regards the proposed case studies, the results bring some open questions whose investigation could be beneficial to both elementary and advanced CA-based models. For instance, how to effectively encode the information on self-replication for the purposes of the evolutionary design? Could the CMRs be adapted in order to provide the capability to optimise the number of states during evolution? Are there any new operators for the EA that would allow optimising the evolutionary process itself? These issues represent ideas for our future research.

References

[1] J. von Neumann, *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.
[2] C. G. Langton, "Studying artificial life with cellular automata," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1–3, pp. 120–149, 1986.
[3] M. Sipper, "Studying artificial life using a simple, general cellular model," *Artif. Life*, vol. 2, no. 1, pp. 1–35, 1994.
[4] F. Peper, J. Lee, S. Adachi, and T. Isokawa, "Cellular nanocomputers: A focused review," *International Journal of Nanotechnology and Molecular Computation*, vol. 1, no. 1, pp. 33–49, 2009.
[5] P. Rosin, A. Adamatzky, and X. Sun, *Cellular Automata in Image Processing and Geometry*. Springer, 2014.
[6] M. Espinola, J. Piedra-Fernandez, R. Ayala, L. Iribarne, and J. Wang, "Contextual and hierarchical classification of satellite images based on cellular automata," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 53, no. 2, pp. 795–809, 2015.
[7] S. Sahu, H. Oono, S. Ghosh, A. Bandyopadhyay, D. Fujita, F. Peper, T. Isokawa, and R. Pati, "On cellular automata rules of molecular arrays," *Natural Computing*, vol. 11, no. 2, pp. 311–321, 2012.
[8] P. Yin, S. Sahu, A. Turberfield, and J. Reif, "Design of autonomous dna cellular automata," in *DNA Computing*, ser. Lecture Notes in Computer Science, A. Carbone and N. Pierce, Eds. Springer Berlin Heidelberg, 2006, vol. 3892, pp. 399–416.
[9] G. C. Sirakoulis, "Parallel application of hybrid dna cellular automata for pseudorandom number generation," *Journal of Cellular Automata*, vol. 11, no. 1, pp. 63–89, 2016.
[10] E. F. Codd, *Cellular Automata*. Academic Press, New York, 1968.
[11] M. Sipper, "Quasi-uniform computation-universal cellular cutomata," in *Advances in Artificial Life, ECAL 1995, Lecture Notes in Computer Science, Vol. 929*. Springer Berlin Heidelberg, 1995, pp. 544–554.
[12] E. R. Berlekamp, J. H. Conway, and R. K. Guy, *Winning Ways for Your Mathematical Plays, 2nd Ed., Volume 4*. A K Peters/CRC Press, 2004.
[13] J.-B. Yuns, "Achieving universal computations on one-dimensional cellular automata," in *Cellular Automata for Research and Industry*, ser. Lecture Notes in Computer Science Volume 6350. Springer, 2010, pp. 660–669.
[14] G. D. Stefano and A. Navarra, "Scintillae: How to approach computing systems by means of cellular automata," in *Cellular Automata for Research and Industry*, ser. Lecture Notes in Computer Science, Vol. 7495. Springer, 2012, pp. 534–543.
[15] P. Rendell, "A fully universal turing machine in Conway's game of life," *Journal of Cellular Automata*, vol. 9, no. 1–2, pp. 19–358, 2013.
[16] C. G. Langton, "Self-reproduction in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1–2, pp. 135–144, 1984.
[17] J. Byl, "Self-reproduction in small cellular automata," *Physica D: Nonlinear Phenomena*, vol. 34, no. 1–2, pp. 295–299, 1989.
[18] J. A. Reggia, S. L. Armentrout, H.-H. Chou, and Y. Peng, "Simple systems that exhibit self-directed replication," *Science*, vol. 259, no. 5099, pp. 1282–1287, 1993.

[19] G. Tempesti, "A new self-reproducing cellular automaton capable of construction and computation," in *Advances in Artificial Life, Proc. 3rd European Conference on Artificial Life*, ser. Lecture Notes in Artificial Intelligence, Vol. 929. Springer, 1995, pp. 555–563.
[20] M. Land and R. K. Belew, "No perfect two-state cellular automata for density classification exists," *Physical Review Letters*, vol. 74, pp. 5148–5150, 1995.
[21] M. S. Capcarrere, M. Sipper, and M. Tomassini, "Two-state, $r = 1$ cellular automaton that classifies density," *Physical Review Letters*, vol. 77, pp. 4969–4971, 1996.
[22] S. Sahoo, P. P. Choudhury, A. Pal, and B. K. Nayak, "Solutions on 1-d and 2-d density classification problem using programmable cellular automata," *Journal of Cellular Automata*, vol. 9, no. 1, pp. 59–88, 2014.
[23] S. Wolfram, *A New Kind of Science*. Champaign IL: Wolfram Media, 2002.
[24] S. Ninagawa, "Solving the parity problem with rule 60 in array size of the power of two," *Journal of Cellular Automata*, vol. 8, no. 3–4, pp. 189–203, 2013.
[25] A. Adamatzky, *Identification of Cellular Automata*. CRC Press, 1994.
[26] L. Bull, I. Lawson, A. Adamatzky, and B. DeLacyCostello, "Towards predicting spatial complexity: A learning classifier system approach to the identification of cellular automata," in *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*. IEEE Computer Society, 2005, pp. 136–141.
[27] A. Adamatzky, "Identification of cellular automata," in *Computational Complexity*, R. A. Meyers, Ed. Springer New York, 2012, pp. 1564–1575.
[28] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.
[29] N. H. Packard, "Adaptation toward the edge of chaos," in *J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, Dynamic Patterns in Complex Systems*. World Scientific, 1988, pp. 293–301.
[30] F. C. Richards, T. P. Meyer, and N. H. Packard, "Extracting cellular automaton rules directly from experimental data," *Physica D*, vol. 45, no. 1–3, pp. 189–202, 1990.
[31] E. Sapin, O. Bailleux, and J. Chabrier, "Research of complexity in cellular automata through evolutionary algorithms," *Complex Systems*, vol. 17, no. 3, pp. 231–241, 2007.
[32] M. Sipper, *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, Vol. 1194*. Berlin: Springer, 1997.
[33] R. Breukelaar and T. Bäck, "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 New York, 2005, pp. 107–114.
[34] M. Mitchell, J. P. Crutchfield, and P. Hraber, "Evolving cellular automata to perform computations: Mechanisms and impediments," *Physica D*, vol. 75, no. 1–3, pp. 361–391, 1994.
[35] E. Sapin, "Gliders and glider guns discovery in cellular automata," in *A. Adamatzky (ed.), Game of Life Cellular Automata*. Springer, 2010, pp. 135–165.
[36] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: Wiley, 1966.
[37] W. Elmenreich and I. Fehérvári, "Evolving self-organizing cellular automata based on neural network genotypes," in *Proceedings of the 5th International Conference on Self-organizing Systems*. Springer, 2011, pp. 16–25.
[38] D. Medernach, T. Kowaliw, C. Ryan, and R. Doursat, "Long-term evolutionary dynamics in heterogeneous cellular automata," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2013, pp. 231–238.
[39] S. Bandini, A. Bonomi, and G. Vizzari, "An analysis of different types and effects of asynchronicity in cellular automata update schemes," *Natural Computing*, vol. 11, no. 2, pp. 277–287, 2012.
[40] M. A. Javaheri Javid, M. M. al Rifaie, and R. Zimmer, "Detecting symmetry in cellular automata generated patterns using swarm intelligence," in *Theory and Practice of Natural Computing*, ser. Lecture Notes in Computer Science, A.-H. Dediu, M. Lozano, and C. Martn-Vide, Eds. Springer International Publishing, 2014, vol. 8890, pp. 83–94.
[41] J. Skaruz, F. Seredynski, and A. Piwonska, "Two-dimensional patterns and images reconstruction with use of cellular automata," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 9–16, 2014.
[42] J. M. Baetens and B. De Baets, "Towards a comprehensive understanding of multi-state cellular automata," in *Cellular Automata*, ser. Lecture Notes in Computer Science, J. Ws, G. C. Sirakoulis, and S. Bandini, Eds. Springer International Publishing, 2014, vol. 8751, pp. 16–24.
[43] J. Baetens and B. De Baets, "Phenomenological study of irregular cellular automata based on Lyapunov exponents and jacobians," *Chaos*, vol. 20, no. 3, p. 15, 2010.
[44] A. Bisio, G. DAriano, P. Perinotti, and A. Tosini, "Free quantum field theory from quantum cellular automata," *Foundations of Physics*, pp. 1–16, 2015.
[45] V. Mardiris, G. Sirakoulis, and I. Karafyllidis, "Automated design architecture for 1-d cellular automata using quantum cellular automata," *Computers, IEEE Transactions on*, vol. 64, no. 9, pp. 2476–2489, 2015.
[46] K. Sridharan and V. Pudi, *Design of Arithmetic Circuits in Quantum Dot Cellular Automata Nanotechnology*. Springer International Publishing Switzerland, 2015.
[47] A. N. Bahar, S. Waheed, and N. Hossain, "A new approach of presenting reversible logic gate in nanoscale," *SpringerOpen Journal - Electronics and Electrical Engineering*, vol. 4, 2015.
[48] S. Sahu, H. Oono, S. Ghosh, A. Bandyopadhyay, D. Fujita, F. Peper, T. Isokawa, and R. Pati, "Molecular implementations of cellular automata," in *Cellular Automata for Research and Industry*, ser. Lecture Notes in Computer Science, Vol. 6350. Springer, 2010, pp. 650–659.
[49] H. Balijepalli and M. Niamat, "Design of a nanoscale quantum-dot cellular automata configurable logic block for fpgas," in *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*. IEEE, Aug 2012, pp. 622–625.
[50] M.-A. Tsompanas, G. Sirakoulis, and A. Adamatzky, "Evolving transport networks with cellular automata models inspired by slime mould," *Cybernetics, IEEE Transactions on*, vol. 45, no. 9, pp. 1887–1899, 2015.
[51] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
[52] D. Andre, F. H. Bennett, III, and J. R. Koza, "Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*. MIT Press, 1996, pp. 3–11.
[53] D. Andre, F. H. Bennett III, and J. R. Koza, "Evolution of intricate long-distance communication signals in cellular automata using genetic programming," in *In Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*. MIT Press, 1996, pp. 513–522.
[54] Y. Zhao and S. A. Billings, "The identification of cellular automata," *Journal of Cellular Automata*, vol. 2, no. 1, pp. 47–65, 2007.
[55] K.-I. Maeda and C. Sakama, "Identifying cellular automata rules," *Journal of Cellular Automata*, vol. 2, no. 1, pp. 1–20, 2007.
[56] M. F. Brameier and W. Banzhaf, *Linear Genetic Programming*. Springer, 2007.
[57] M. Bidlo and Z. Vasicek, "Evolution of cellular automata using instruction-based approach," in *2012 IEEE World Congress on Computational Intelligence*. IEEE Computer Society, 2012, pp. 1060–1067.
[58] ——, "Evolution of cellular automata with conditionally matching rules," in *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*. IEEE Computer Society, 2013, pp. 1178–1185.
[59] M. Bidlo, "Evolving multiplication as emergent behavior in cellular automata using conditionally matching rules," in *2014 IEEE Congress on Evolutionary Computation*. IEEE Computational Intelligence Society, 2014, pp. 2001–2008.
[60] J.-Y. Perrier, M. Sipper, and J. Zahnd, "Toward a viable, self-reproducing universal computer," *Physica D*, vol. 97, pp. 335–352, 1996.

Michal Bidlo received the Ph.D. degree in information technology from the Faculty of Information Technology (FIT), Brno University of Technology (BUT), Czech Republic, in 2009. He is an Assistant Professor at the Department of Computer Systems FIT BUT. His research interests include cellular automata, evolutionary computation, evolvable hardware and bio-inspired systems. Author or co-author of over 20 conference/journal papers focused on evolutionary design and evolvable hardware.