Investigation of Replicating Tiles in Cellular Automata Designed by Evolution Using Conditionally Matching Rules

Michal Bidlo
Brno University of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2, 61266 Brno, Czech Republic
Email: bidlom@fit.vutbr.cz

***Abstract*—In this paper we investigate the evolutionary design of replicating tiles in cellular automata. In particular, various sizes of the tiles will be considered whose replication ought to be performed by satisfying a given arrangement of the tiles with respect to each other. The goal is to determine the abilities of the genetic algorithm in combination with conditionally matching rules used for representing the transition functions of cellular automata to find solutions for tiles consisting of up to a hundred of cells. A comparative study will be presented considering the success rate, computational effort and complexity of the obtained solutions as the main values of interest. It will be shown that, in addition to the tile size and the number of states of the cellular automaton, the probability of finding a correct solution is also substantially influenced by the arrangement style. The results show that the tile arrangement that may be considered as the simplest one does not have to necessarily be easily realisable by the genetic algorithm as a transition function for a cellular automaton.**

I. INTRODUCTION

Self-organisation, replication and emergent behaviour in general represent some of the popular issues often studied with respect to systems which behaviour is typically spatially-controlled (distributed) without any centralised control. As the systems become more complex, these processes may represent the only reasonable way of achieving a required state or conditions for a correct operation. These phenomena are important, among others, in nanotechnology, e.g. for designing new materials with some specific properties, molecular systems etc. [1]. Moreover, a study has been published demonstrating a platform that utilizes self-organization in order to perform some kinds of computation [2]. The area of potential applications is thus wide and the research of those techniques may become increasingly important in the near future.

Cellular automata (CA) represent a simple uniform platform that provided a convenient way for the investigation of the behaviour of complex systems. Since their introduction by Ulam and von Neumann in 1966 [3] where self-reproduction of computing structures was one of the primary goal researchers have dealt, among others, how to effectively design a cellular automaton (and its transition function in particular) to solve specific tasks. Some of the most popular applications of CA include the famous Conway's Game of Life [4], simulation of various concepts of universal computation [5][6] or modelling physical and biological systems [7][8].

The problem of self-replication was investigated using various structures (usually known as replicating loops). Langton's loop [9] or some simplified variants like Byl's loop [10] or a loop of Chou and Reggia [11] probably represent the most known instances. The replication task basically involves a given structure that is able to make its identical copies filling out the cellular space in a well-organised way. Although the original Langton's loop utilizes some concepts from von Neumann's universal constructor [3] (especially a construction "arm" used to develop a copy of the loop according to some "signal" states inside it), the CA development results in a regular grid of static structures tessellating the cellular array. The latter works tried to simplify this process while preserving the original (abstract) Langton's concept of the replication control. The result is that the CA is tessellated by the given structures (some of them can even exhibit dynamic behaviour [10]) whose organisation in the cellular array is very similar.

In addition to the self-replication a growth of self-organising structures in cellular automata has also been widely studied both from theoretical and application perspective. In such case the CA works as a generator of specific structures (patterns) with some given properties or abilities. Some transition functions exist for one-dimensional (1D) CA whose development produces a sequence of states that can be, for instance, interpreted as pseudo-random numbers, Turing machine simulation, square calculation or prime numbers [8][12]. In [13] Basanta et al. used a genetic algorithm to evolve the rules of effector automata (a generalised variant of CA) to create microstructural patterns (similar to crystal structures known from some materials). An important aspect of this work was to investigate new materials with specific properties and their simulation using computers. Suzudo proposed an approach to the evolutionary design of 2D asynchronous CA for a specific formation of patterns in groups in order to better understand of the pattern-forming processes known from nature [14]. Elmenreich et al. proposed an original technique for growing self-organising structures in CA whose development is controlled by neural networks according to the internal cell states [15].

In general the study of the cellular automata behaviour represents an important issue in the area of complex sys-

tems. Considering the uniform structure of the CA as an architecture potentially suitable for some future technologies, the understanding of their capabilities and functioning at the elementary level may become crucial for successful applications. Therefore, the research of efficient approaches to the design (programming) of cellular systems and investigating their features related to the emergent behaviour can provide worthwhile pieces of knowledge both from the theoretical and practical point of view.

A new approach for representing the transition functions of the CA for the purposes of the evolutionary design called conditionally matching rules (CMR) was proposed in [16]. This method showed as very promising even for the design of complex cellular automata (e.g. see [17]). The experiments showed that using the CMR approach in combination with a genetic algorithm is able to provide solutions for some problems in CA for which the conventional (table-based) representation of the transition function failed.

For the purposes of this paper the CA will be investigated with respect to their ability to replicate some specific tile-like structures of various sizes and target arrangements. The goal is to demonstrate that such CA can be discovered automatically using a genetic algorithm in combination with the conditionally matching rules. One of the key factors of this study is the aim to discover various tessellation scenarios for squared tiles as large as possible. The tessellation of the cellular array by the tiles is performed through the process of the tile replication. It will be shown that the success of finding working tessellation rules for a given tile strongly depends, in addition to the tile size, also on the required tessellation style.

II. FUNDAMENTALS OF CELLULAR AUTOMATA

The original concept of cellular automaton introduced in [3], that will be considered in this paper, assumes a 2D matrix of cells, each of which at a given moment acquires a state from a finite set of states. The development of the CA is performed synchronously in discrete time steps by updating the cell states according to local transition functions of the cells. Uniform cellular automata will be considered in which the local transition function is identical for all cells and will be denominated as a transition function of the CA. The state of a cell in a subsequent time step depends on the combination of states in the cell neighbourhood. In this paper von Neumann neighbourhood will be assumed that includes the central (C) cell to be updated and its immediate neighbours in the north (N), south (S), east (E) and west (W) direction (i.e. the neighbourhood of each cell consists of 5 cells in total).

Since the CA behaviour can be practically evaluated in a finite-size cellular array, boundary conditions need to be specified in order to correctly determine cell states at the edge of the array. In this paper cyclic boundary conditions will be implemented which means that the cells at an edge of the CA are "connected" with the appropriate cells on the opposite edge (i.e. these cells are considered as neighbours) in each dimension. In case of the 2D CA the shape of the cellular array can be viewed as a toroid.

The transition function is usually defined as a mapping that for all possible combinations of states in the cellular neighbourhood determines a new state. This mapping can be represented as a set of rules of the form $N\ W\ C\ E\ S \rightarrow C_{t+1}$ where N, W, C, E and S denote cell states in the defined neighbourhood at a time t and C_{t+1} is the new state of the cell in the middle of the neighbourhood. It means that for every possible combination of states $N\ W\ C\ E\ S$ a new state C_{t+1} needs to be determined. However, if the number of cell states increases (typically in non-binary, i.e. multi-state CA), the number of possible transition rules grows exponentially which is inconvenient for efficient CA design. Of course, not all transition rules need to be specified explicitly but the problem is how to choose the rules which modify the central cell in the neighbourhood. In order to overcome this issue, advanced representation was proposed in [16] and denominated as Conditionally Matching Rules that allows to reduce the size of representation of the transition functions. This approach will be considered in this paper for the purposes of the evolutionary design of cellular automata.

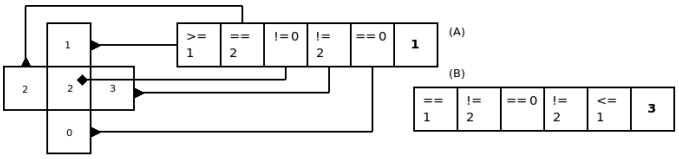

Fig. 1. Example of a conditionally matching rule specified for 5-cell neighbourhood. The value of the new state is written in bold. (A) example of a matching CMR, (B) example of a CMR that does not match – the second and third condition is evaluated as false.

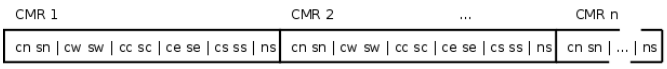

Fig. 2. Structure of a chromosome for genetic algorithm encoding a CMR-based transition function. cx denote a condition for the cell at position x in the neighbourhood, sx represents the state value to be investigated using the appropriate condition with respect to the state of cell at position x, ns specifies the next state for a given CMR. All the conditions and state values are represented by integer numbers.

III. CONDITIONALLY MATCHING RULES

The concept of conditionally matching rules showed as a very promising technique in comparison with the conventional table-based approach considering various experiments with binary cellular automata (e.g. in [16]) as well as more complex multi-state CA [17].

A conditionally matching rule represents a generalized rule of a transition function for determining a new cell state. Whilst the common approach specifies a new state for each specific combination of states in the cellular neighbourhood, the CMR allows to specify a wider range of combinations in a single rule. 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 a 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 state of a given 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 evaluation is true, then the condition item is said to match

with the state of the appropriate cell in the neighbourhood. In order to be able to determine the new state according to a given CMR, all its condition items must match (in such case the CMR is said to match).

The following condition functions will be considered: $==0, \neq 0, \leq, \geq$. Note that this condition set represents a result of our long-term experimentation and experience with the CMR approach and will be used for all the experiments in this paper. The first two conditions $== 0$, respective $\neq 0$ evaluates whether the corresponding cell state is equal to 0 (i.e. a "dead" state), respective whether it is different from state 0. Note that the state value of the condition item for $== 0$ and $\neq 0$ is considered implicitly within the condition itself. The remaining two conditions represent relational operators "less or equal" and "greater or equal" for which the state value of the condition item must be explicitly specified.

Figure 1 shows an example of conditionally matching rules define for a 2D CA with 5-cell neighbourhood together with illustration of cells the condition entities are related to. CMR (A) is a matching CMR since all the conditions of its conditional part are evaluated as true with respect to the sample neighbourhood. On the other hand, CMR (B) does not match because the second condition item $! = 2$ evaluates as false with respect to the west cell that possesses state 2. Similarly, the third condition $== 0$ is not true as the central cell is in state 2.

A CMR-based transition function can be specified as a finite sequence of conditionally matching rules. The following algorithm will be applied to determine a new state of a cell. The CMRs are evaluated sequentially one by one. The first matching CMR in the sequence is used to determine the new state. If no of the CMRs matches, then the cell keeps its current state. The conventions for evaluating and applying the CMRs ensure that the process of calculating the new state is deterministic (considering an assumption that the condition functions are deterministic too). Therefore, it is possible to convert the CMR-based transition function to a corresponding table-based representation, preserving the fundamental concept of cellular automata. Moreover, every condition set that includes relation $==$ allows to formulate transition rules for a specific combination of states if needed (by specifying $==$ as a condition for all items of the CMR similarly to the table-based approach).

In order to obtain the conventional representation of the transition function from an evolved CMR solution, the following algorithm is applied using the same CA that was considered during evolution. Let C_t and C_{t+1} denote states of a cell in two successive steps at the time t and $t+1$ respectively. A transition rule of the form $N_t\ W_t\ C_t\ E_t\ S_t \rightarrow C_{t+1}$ is generated for a given combination of states of the cellular neighbourhood if $C_t \neq C_{t+1}$. This process is performed after each step and for each cell until the CA reaches a stable or periodic state. The set of (table-based) rules obtained from this transformation process represents the corresponding conventional prescription of the transition function.

IV. EVOLUTIONARY SYSTEM SETUP

A simple genetic algorithm (GA) was utilized for the evolution of CMR-based transition functions in order to achieve a given behaviour in cellular automata. Each chromosome of the GA represents a candidate transition function encoded as a finite sequence of conditionally matching rules. The chromosome is implemented as a vector of integers encoding the condition items and next states of the CMRs. The length of the chromosomes is given by the number of CMRs which is specified as a parameter for a specific experiment. The structure of a chromosome is shown in Figure 2.

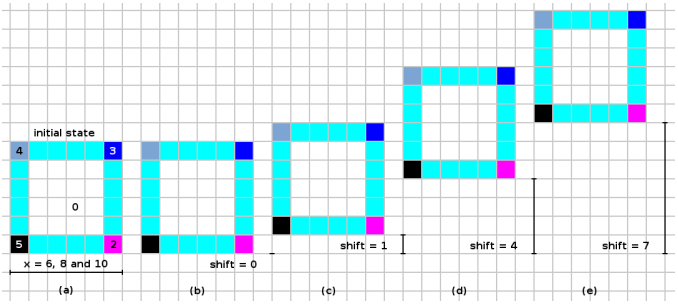

Fig. 3. Structure of the tile and tessellation arrangements considered in this paper: (a) Various sizes of the (squared) tile is considered which is given by the parameter x. A single tile represents the initial CA state. (b)-(e) Various tessellation styles are specified by a $shift$ parameter determining a required arrangement of the target (replicated) tile with respect to the initial tile. A line of cells in state 0 is required to separate the tiles.

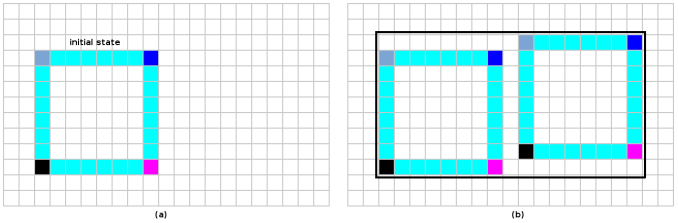

Fig. 4. Example of pattern transformation considering 8x8-cell tile whose replica ought to be shifted by 1 cell upward: (a) initial CA state, (b) target pattern (the thick rectangle represents the smallest bounding box of the initial tile and its copy.

The population of the GA consists of 8 chromosomes that are initialized randomly at the beginning of the evolutionary process. In each generation four individuals are selected randomly from the current population the best one of which is considered as a parent (i.e. it is a case of tournament selection with the base 4). In order to generate an offspring, the parent undergoes a process of mutation as follows. A random integer M in range from 0 to 2 is generated. Then M random positions within the parent chromosome are selected. The offspring is created by replacing the original integers at these positions by new randomly generated values. If M equals 0, the mutation is not performed and the offspring is identical to the parent. The selection and mutation is repeated until a whole new population is created. Crossover is not applied because the experiments showed no significant benefit of this operator with respect to the obtained results.

Several sets of experiments were performed considering various numbers of CMRs, sizes of the tiles and tessellation styles. In particular, the tile is considered as a squared circuit of cells in state 1 whose corners are represented by cells in states 2, 3, 4 and 5 (see Figure 3a). Various sizes of the tile are considered, a specific size is given by a parameter x that

determines the number of cells of the tile side (i.e. for the squared tiles the area occupied by the tile is $x \times x$ cells). A tile of a specific size is considered as initial CA state. The goal of each experiment is to find suitable transition rules of the CA according to which the tile will replicate with respect to a required tessellation style. The tessellation style specifies the arrangement of the tile replica with respect to the initial state or the previous replica (let us call it a predecessor). In all cases the replica is required to emerge on the right of the predecessor and separated by a single "column" of cells in state 0. Various tessellation styles are considered in which the tile replica is shifted upward by a given number of cells against its predecessor. The number of cells of the shift is specified as a parameter that determines vertical arrangement of the tiles as shown in Figure 3b-e.

The following fitness function is considered to evaluate the candidate solutions. A partial fitness is defined after each CA step as the number of cells in correct state. A final fitness is defined as the maximum from all the partial fitness values for a given number of CA steps. The tile replication is thus considered as a pattern transformation problem developing a tile replica from the original (initial) tile. The CA state containing the initial tile and its copy (satisfying the required arrangement given by the $shift$ value) is considered as a target pattern. For example, if 8x8-cell tile is considered with the $shift$ parameter of value 1, then the CA initial state corresponds to the pattern shown in Figure 4a and the target state developed after a finite CA steps contains the original tile and its replica on the right shifted by 1 cell upward as illustrated in Figure 4b. Note that the CA size is chosen by two cells larger on each side with respect to the target state marked by a thick rectangle in order to allow to directly evaluate the target pattern and not to strictly limit the tile development by the CA boundary conditions. It means that for the example from Figure 4 the maximal fitness value (given by the CA size) corresponds to $F_{max} = 13 \times 21 = 273$. Although only a single copy of the tile is required during the evolution, the main goal is to search for such solutions that are able to produce more copies tessellating the cellular space with respect to the given parameters. Therefore, the obtained results need to be verified in larger CA in order to determine which of them are general. For the purposes of this paper all solutions that are able to develop more than two copies of the original tile are considered as general.

Each set of experiments is performed for a specific setup of the following parameters: the number of CMRs (considered for the values 20, 30, 40 and 50), the size of the tile x (considered sizes are 6, 8 and 10), the number of cell states (the CA working with 6, 8 and 10 states) and the shift of the tiles (considered by 0, 1, 4 and 7 cells). In order to evaluate the fitness, the CA develops for 40 steps for the tile size $x = 10$, for the smaller tiles (if x is 8 or 6) the CA develops for 30 steps. For each setup (combination of the aforementioned values) 100 independent evolutionary runs are performed. The genetic algorithm is executed for a maximum 2 millions of generations. If no correct solution is found within this generation limit, the evolution is terminated. The experiments were executed using the Anselm cluster[1], the execution time of a single evolutionary run was from 6 to 12 hours depending on the size of the tile (CA). The results were analysed regarding the success rate, computational effort and quality of the solutions generated by the CA.

N	W	C	E	S	Cnew	N	W	C	E	S	Cnew	N	W	C	E	S	Cnew	N	W	C	E	S	Cnew
0	0	0	0	5	4	0	1	4	1	4	0	1	0	0	1	3	1	4	0	0	1	1	1
0	0	1	0	0	5	0	1	4	2	0	1	1	0	1	1	0	0	4	0	0	4	0	3
0	0	1	0	4	0	0	1	4	3	0	1	1	0	2	0	0	5	4	0	1	1	0	0
0	0	1	1	0	0	0	1	4	3	3	1	1	0	4	5	0	0	4	0	1	4	0	0
0	0	1	1	3	0	0	1	4	4	2	3	1	0	5	1	3	1	4	0	3	5	0	0
0	0	1	3	0	0	0	1	4	4	5	1	1	1	0	0	1	1	4	0	4	1	0	0
0	0	1	4	0	0	0	3	0	0	0	4	1	1	0	2	2	1	4	0	4	1	1	1
0	0	1	4	4	0	0	3	0	0	1	3	1	1	1	1	0	0	4	0	5	0	0	1
0	0	3	1	0	0	0	3	1	0	0	4	1	1	1	5	0	0	4	0	5	1	0	3
0	0	4	0	0	5	0	3	1	1	2	4	1	1	2	2	0	0	4	1	0	0	0	2
0	0	4	0	4	5	0	3	4	0	0	5	1	1	4	0	1	1	4	1	0	1	0	3
0	0	4	0	5	5	0	3	4	1	1	3	1	1	4	2	2	1	4	1	2	0	0	0
0	0	4	1	0	0	0	4	0	0	0	1	1	1	5	1	2	1	4	1	5	0	0	0
0	0	4	4	5	0	0	4	0	0	1	1	1	3	0	1	3	3	4	2	0	0	0	2
0	0	5	1	0	1	0	4	0	0	4	3	1	3	1	1	1	4	4	2	2	0	0	0
0	0	5	1	1	4	0	4	0	1	0	3	1	3	1	4	1	4	4	3	5	1	5	1
0	0	5	1	3	0	0	4	0	5	0	3	1	3	3	1	0	0	4	4	5	0	0	0
0	1	1	0	0	4	0	4	1	1	1	4	1	3	4	1	0	0	4	5	0	0	0	2
0	1	1	0	1	4	0	4	3	0	5	1	1	3	4	1	1	1	4	5	2	0	0	0
0	1	1	1	1	4	0	4	3	1	0	1	1	4	0	0	0	2	4	5	4	0	0	0
0	1	1	1	2	4	0	4	4	0	0	1	1	4	0	2	0	4	5	0	0	1	1	1
0	1	1	4	1	4	0	4	4	1	2	3	1	5	0	0	0	2	5	0	1	1	0	0
0	1	2	0	0	4	0	4	4	3	2	3	1	5	1	5	0	4	5	0	3	1	0	0
0	1	3	1	0	1	0	5	0	0	0	1	3	0	1	1	0	0	5	1	0	0	0	2
0	1	3	3	0	1	0	5	0	4	0	3	3	1	1	1	0	0	5	1	2	0	0	5
0	1	4	1	0	1	0	5	3	0	0	1	3	3	5	1	3	0	5	1	5	0	0	0
0	1	4	1	1	3	1	0	0	0	4	1	3	5	0	2	0	1	5	4	0	0	0	2
0	1	4	1	2	3	1	0	0	1	1	1	4	0	0	1	0	3	5	4	2	0	0	5

Fig. 5. One of the best transition functions obtained for the 10×10-cell tile whose arrangement is given by the parameter $shift = 7$. It consists of 112 rules and the tile replicates in 40 steps. This solution was evaluated as general.

V. EXPERIMENTAL RESULTS

The experiments provided successful solutions for most setups mentioned in Section IV. Several general solutions were discovered for all considered sizes of the tiles. Table I summarises the statistical results of all the considered evolutionary experiments. Although the solutions of the tessellating tiles were required to be static (i.e. a finished tile should not longer change during the CA development), some general results were identified that are able to generate more complex tile behaviour that fulfils the fitness specification too. For example, the tile possesses a blinking cell or a small part of the tile changes periodically for several steps producing a dynamic effect that does not lead to the tile destruction. These solutions are described by "with a blinking cell" or "partially dynamic" in Table I.

The results show that a wide variety of tile replication processes can be discovered to tessellate the cellular space. It is indicated by the minimal CA steps needed to produce the target pattern in comparison with the maximal value of this parameter that was specified as 40 steps for the 10×10-cell tiles and 30 steps for smaller tiles. The analysis of the results showed that the number of steps needed to produce a tile copy varies for the obtained solutions from the minimal number for a given experiment up to the maximal value. It can be seen that the number of obtained solutions (i.e. the success rate) decreases with the tile size which is expectable because the transformation process of a large structure can not be considered as a trivial task. The 10×10-cell tile represents

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

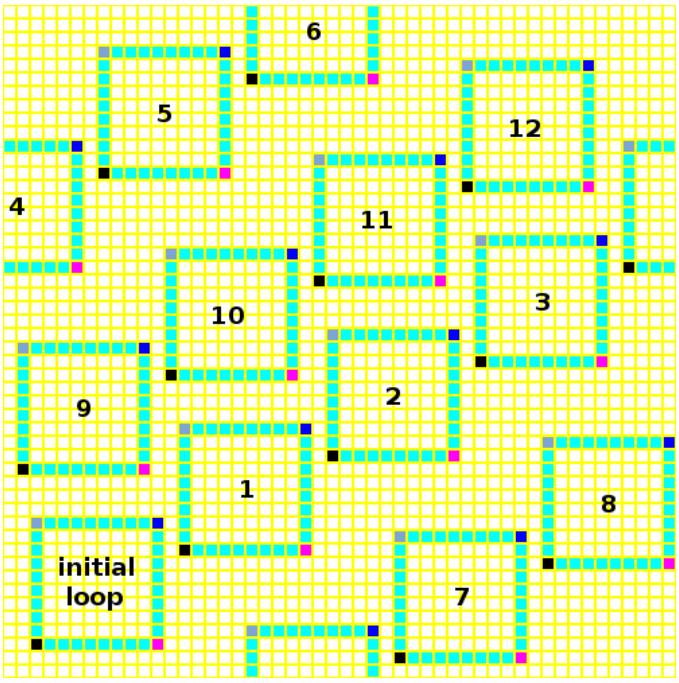

Fig. 7. Example of a general tessellation by means of a 10×10-cell tile according to the transition function from Figure 5. The development took 480 steps in 50x50-cell CA using cyclic boundary conditions. The numbers in tiles indicate their order of creation.

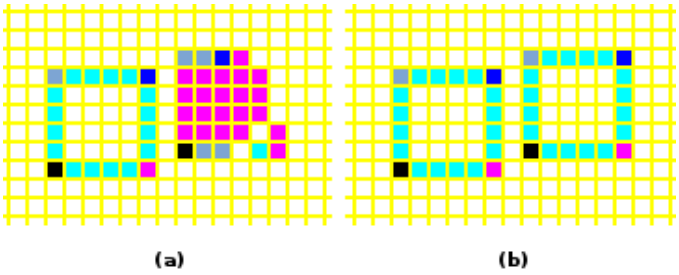

Fig. 8. Example a CA state from a 6×6-cell tile development in which state changes can be observed over the whole tile area.(a) The 13th step showing the tile under development, (b) the 21st steps after which the tile is finished. The CA works with 6 cell states and its transition function consists of 70 effective rules. This solution was evaluated as general.

the largest structure of this type for which a working solution was obtained so far in our sets of experiments. An interesting phenomenon can be observed considering the scenarios with various values of the $shift$ parameters. The results from Table I show that the highest success rate was achieved for $shift = 4$. In this case the highest number of general solutions can mostly be identified (especially for the smallest tile considered consisting of 6x6 cells). Surprisingly, the scenario with $shift = 0$ (which can be visually considered as the simplest tile arrangement) provided only 4 general solutions in total which is the lowest value out of all the tessellation styles if the $shift$ parameter is considered. A possible explanation of this fact may be in a complexity of the process transforming the corners of the original tile onto its copy when the top and bottom corners of both tiles lie at the same levels (see Figure 3a-b). Some of the potential solutions to overcome this issue may be to increase the number of CMRs, the maximal number of CA steps or to provide more evolution time. However, the problem of scale would probably represent a major obstacle for obtaining more successful solutions in a reasonable time. Considering the number of cell states it can be observed that in some cases more cell states provide slightly higher success rate or the number of general solutions for a given tile size. It is understandable because more cell states allow more rules to be available for the tile transformation process, on the other hand, however, an increase of the cell states causes an exponential growth of the solution space needed to be explored by the GA (the problem of scale). Therefore, the increase of the success rate is rather low for the same generation limit.

In order to demonstrate some of the successful results in more detail, the 10×10-cell tiles will be considered in CA working with 10 cell states and $shift = 7$ which represents one of the most complex setups investigated in this paper. Figure 5 shows a transition function (in particular its table form generated from the corresponding evolved CMR-based representation) for the replication of the tile. The complete CA development of a single replica from the initial state is shown in Figure 6. The CA needs 40 steps in order to finish the replication process which is the maximal value allowed for this experiment. The transition function consists of 112 effective rules and represents one of the best solutions obtained in this paper. As evident from Figure 6 the transformation process starts from the upper-right corner which is probably the simplest solution with respect to the fact that the replica ought to be shifted 7 cells upwards against the original tile. The development also shows that after the 10th step the replica is created by a parallel construction of the left and bottom edge of the tile and continues with the right and upper edge that "close" the complete tile after the 40th step. More results can be identified that construct the tiles by "following" its edges which is actually a dominant approach evolved for the successful solutions of larger tiles. One of the other solutions for the 10×10-cell tile was found whose development takes 36 steps to finish the replica, its transition function contains 815 effective rules. This is the best result considering the replication speed, however, the transition functions is several times more complex than that from Figure 5. For smaller tiles (especially 6×6-cell instances) the development often utilizes also the "inner" part of the tile where the states are altered during the tile construction and then progressively "cleared" by setting states 0 before the replica is finished (see an example state from a 6×6-cell tile development in Figure 8). Although the results are able to develop tiles with a required (regular) arrangement, the process of construction from the initial state appears to be rather chaotic so it is difficult to manually alter or optimize the transition function. Note that all the mentioned results were evaluated as general, i.e. the CA is able to produce more tile replicas with the given arrangement (given by the $shift$ parameter) if the development continues. Figure 7 demonstrates a tessellation of 10×10-cell tiles according to the transition function from Figure 5. It is a case of ongoing development of the tile from Figure 6 if a larger CA is considered which shows the ability to successfully perform the replication for the situation that was not explicitly considered in the fitness evaluation performed during the evolutionary process (i.e. for more than 40 steps of the CA).

Overall the results obtained from the experiments showed an ability of the CA to perform a wide variety of transformation processes that can be interpreted as a (general) replication of tiles of given dimensions and shape. Although no other specific operation was required in addition to the copying te tiles, the information provided by the replication process give rise of some open questions related to various aspects of cellular automata. For example, could some of these processes be utilized practically? For instance, the concept of tessellation of an area by a specific entity may be interesting in nanotechnology and artificial molecular systems to create (or simulate the creation of) a surface with given properties. Another view of the results may consider performing computations (or more general information processing) using a special encoding by means of cell states inside (or along with) the transforming structures. For example, a concept of self-replicating structures capable of computation were presented in [18] which indicate that more similar techniques could be possible using similar structures and suitable encoding implemented in CA. Moreover, because of a primary interest in uniform cellular automata, their physical realisation would be feasible.

VI. CONCLUSIONS

This paper investigated automatic evolutionary design of cellular automata in the problem of replication of tile-like structures with some predefined arrangements in the cellular array. The goal was to automatically dioscover suitable transition functions for the CA using genetic algorithm and Conditionally Matching Rules and to determine some interesting aspects of the design process and obtained results depending on the setup parameters.

The results showed that it is possible to design general replication functions for each of the considered tile sizes in most sets of experiments. Although the discovery of the transition functions represents a difficult task especially for larger tiles, the evolution provided a wide variety of algorithms that are able to successfully replicate the loops in CA satisfying the given parameters. It was shown that the arrangement of the tiles significantly influence both the success rate of the evolution and the probability of discovering general solutions. Specifically, the design of a CA for the replication of tiles lying in the same horizontal "level" in the cellular array showed to be much more difficult than if the tile replicas of the same size are required to shift vertically by several cells with respect to each other. This is an interesting observation because the tiles in the same level can be considered as the simplest tessellation style. Nevertheless, the successful experiments provided some results for most of the considered setups. Note that the solutions obtained for the 10×10-cell tiles in 10-state CA still represent the largest and most complex structures of this type for which transition functions were automatically designed by the evolution.

The number of various solutions discovered in the experiments indicates that there may be a big potential of the CA for utilization of similar processes in some specific practical applications. The representation of the transition functions by means of conditionally matching rules in combination with the genetic algorithm still appears as an efficient tool to automatically design cellular automata. Therefore, the research will continue with investigating other specific processes and algorithms that the CA may implement (including computations using some unconventional encodings, simulation of biological or physical phenomena and so on).

ACKNOWLEDGMENT

This work was supported by the Czech Science Foundation via the project no. GA14-04197S *Advanced Methods for Evolutionary Design of Complex Digital Circuits*, the BUT project no. FIT-S-14-2297 *Architecture of parallel and embedded computer systems* and the *IT4Innovations Centre of Excellence*, no. CZ.1.05/1.1.00/02.0070, funded by the European Regional Development Fund and the national budget of the Czech Republic via the Research and Development for Innovations Operational Programme.

REFERENCES

[1] B. B. (Ed.), *Springer Handbook of Nanotechnology*. Springer, 2010.

[2] G. S. Snider, "Self-organized computation with unreliable, memristive nanodevices," *Nanotechnology*, vol. 18, no. 36, 2007.

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

[4] 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.

[5] M. Sipper, *Evolution of Parallel Cellular Machines – The Cellular Programming Approach, Lecture Notes in Computer Science, volume 1194*. Berlin: Springer-Verlag, 1997.

[6] A. Ilachinski, *Cellular Automata: A Discrete Universe*. World Scientific, 2001.

[7] D. Griffeath and C. M. (Eds.), *New Constructions in Cellular Automata*. Oxford University Press, 2003.

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

[9] C. G. Langton, "Self-reproduction in cellular automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1–2, pp. 135–144, 1984.

[10] J. Byl, "Self-reproduction in small cellular automata," *Physica D: Nonlinear Phenomena*, vol. 34, no. 1–2, pp. 295–299, 1989.

[11] 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.

[12] J. Kari, "Universal pattern generation by cellular automata," *Theoretical Computer Science*, vol. 429, pp. 180–184, 2012.

[13] D. Basanta, P. Bentley, M. Miodownik, and E. Holm, "Evolving cellular automata to grow microstructures," in *Genetic Programming*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2610, pp. 1–10.

[14] T. Suzudo, "Searching for pattern-forming asynchronous cellular automata – an evolutionary approach," in *Cellular Automata*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3305, pp. 151–160.

[15] 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-Verlag, 2011, pp. 16–25.

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

[17] M. Bidlo, "On routine evolution of new replicating structures in cellular automata," in *7th International Conference on Evolutionary Computation Theory and Applications*. SCITEPRESS, 2015.

[18] 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 Verlag, 1995, pp. 555–563.

TABLE I. STATISTICAL RESULTS OF THE EVOLUTIONARY EXPERIMENTS CONSIDERING THE DESIGN OF CELLULAR AUTOMATA TESSELLATING THE TILES ACCORDING TO THE SCHEME FROM FIGURE 3. THE FOLLOWING INDICATORS WERE MEASURED: SUCC. RATE – THE NUMBER OF SUCCESSFUL EXPERIMENTS OUT OF 100 INDEPENDENT RUNS, AVG. #GEN – THE AVERAGE NUMBER OF GENERATIONS NEEDED TO EVOLVE A WORKING SOLUTION, MIN. STEPS – THE MINIMAL NUMBER OF STEPS OF THE CA NEEDED TO DEVELOP A TARGET PATTERN, MIN. RULES – THE MINIMAL NUMBER OF TABLE-BASED TRANSITION RULES GENERATED FROM THE EVOLVED CMRS THAT ARE ABLE TO DEVELOP THE TARGET PATTERN, #GENERAL – THE NUMBER OF GENERAL SOLUTIONS OUT OF THE SUCCESSFUL RESULTS IN A GIVEN SET OF EXPERIMENTS.

Tile: 6x6, 6-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	19	1118918	18	40	10	1437378	18	52	5	1450986	22	74	0	-	-	-
30	9	1107742	14	47	19	928397	16	38	23	1082634	18	58	2	771103	28	143
40	3	1505039	14	82	11	1005075	18	70	22	1104610	17	60	2	621280	28	143
50	8	1254517	14	58	6	1229602	20	105	18	1085582	23	60	9	1192477	24	110
#general		0				17				30				0		

Tile: 6x6, 8-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	4	998526	19	66	10	1207402	16	66	7	1195744	23	41	0	-	-	-
30	7	1179472	13	45	17	1041444	17	57	24	1081683	18	60	0	-	-	-
40	12	704103	14	81	7	827102	16	95	27	1067662	18	59	2	845255	26	224
50	4	1368359	14	189	9	1079094	20	76	23	1132952	18	88	3	577865	28	161
#general		1 with a blinking cell				21				38				1 partially dynamic		

Tile: 6x6, 10-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	9	1120262	18	53	4	1390380	21	55	8	1097559	17	51	0	-	-	-
30	3	576642	20	58	11	1156451	17	56	13	1389473	19	57	0	-	-	-
40	9	1089556	14	43	19	1215044	18	60	20	1101590	17	54	1	591975	30	276
50	9	1309687	14	76	6	886018	28	68	28	1012288	18	71	1	438305	30	375
#general		1				20				34				0		

Tile: 8x8, 6-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	6	1086616	24	33	0	-	-	-	0	-	-	-	0	-	-	-
30	5	785931	24	41	1	1751265	26	298	1	449621	30	187	1	1012185	30	56
40	10	1256073	24	63	5	1100859	24	86	0	-	-	-	2	1050160	27	70
50	2	697325	24	234	2	1394366	22	119	3	1780271	28	98	1	1687649	30	243
#general		0				0				1 with a blinking cell				0		

Tile: 8x8, 8-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	7	850749	19	41	3	1484130	20	83	0	-	-	-	0	-	-	-
30	4	1256364	28	53	4	1365843	20	31	2	1489461	28	102	1	1338644	30	77
40	7	1028698	18	48	1	376047	30	325	4	1384559	26	127	0	-	-	-
50	9	950239	24	42	2	670553	30	136	6	1340406	25	52	1	1320655	27	135
#general		2 partially dynamic				2				7				1 with a blinking cell		

Tile: 8x8, 10-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	4	1732212	26	49	1	1720045	25	55	0	-	-	-	0	-	-	-
30	6	1172231	29	46	4	1081762	18	81	2	1931238	26	85	1	889316	30	357
40	7	1414253	22	44	3	811286	24	63	4	1354683	26	126	1	713803	30	156
50	4	1083095	18	88	1	751772	21	104	6	1248443	22	79	1	648766	28	119
#general		2				1 partially dynamic				8				0		

Tile: 10×10, 6-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	11	932021	34	42	1	1720787	39	116	0	-	-	-	0	-	-	-
30	6	755866	28	45	1	617867	34	93	0	-	-	-	3	1322751	36	86
40	3	964643	30	208	2	1113080	36	57	1	1928196	36	318	3	1155481	36	138
50	4	1383659	22	121	5	655085	30	128	0	-	-	-	3	1461295	40	101
#general		0				1 partially dynamic				0				2		

Tile: 10×10, 8-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	12	1094858	33	47	1	1092327	36	68	0	-	-	-	0	-	-	-
30	13	734892	31	42	8	1175000	24	53	0	-	-	-	2	1224878	32	110
40	11	742511	33	43	1	476685	40	159	2	1351586	34	88	2	883498	28	158
50	5	939542	22	49	1	859651	34	79	1	1317299	39	133	2	1217523	38	139
#general		0				0				0				2		

Tile: 10×10, 10-state CA

$shift \rightarrow$	0				1				4				7			
#CMRs	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules	succ. rate	avg. #gen	min. #steps	min. #rules
20	7	1410195	34	43	0	-	-	-	0	-	-	-	1	1993618	30	58
30	8	1358227	36	51	6	1226543	30	64	1	734366	40	124	4	1575214	37	105
40	9	1031719	36	60	3	1263793	26	104	1	1527645	32	123	0	-	-	-
50	6	725987	22	58	4	909644	36	59	0	-	-	-	3	758387	33	94
#general		0				0				2				0		

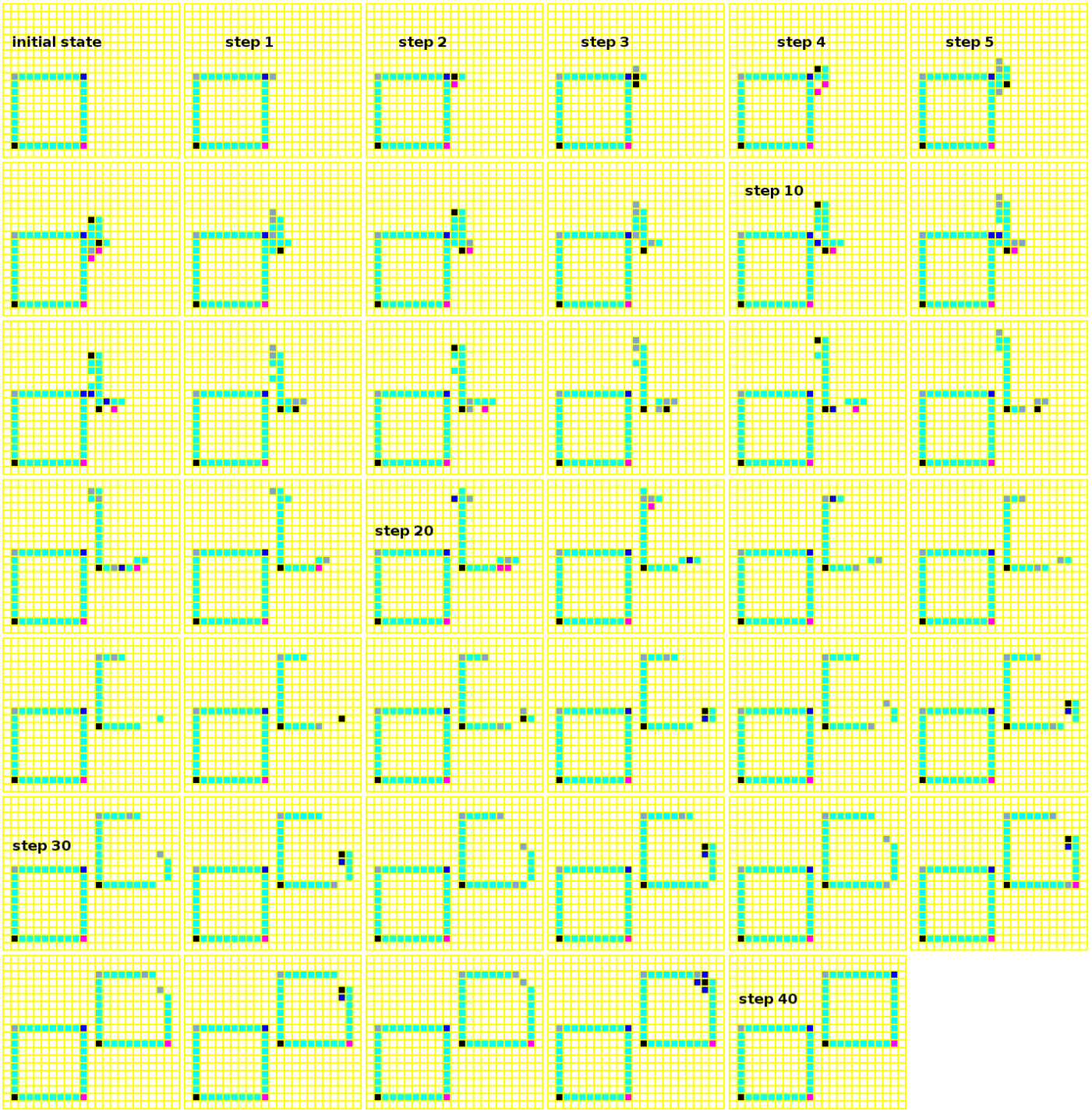

Fig. 6. Development of a 10×10-cell tile using the transition function from Figure 5. The sequence of steps reads from left to right and top to bottom.