Generating Maximal Domino Patterns by Cellular Automata Agents

Rolf Hoffmann[1] and Dominique Désérable[2](✉)

[1] Technische Universität Darmstadt, Darmstadt, Germany
hoffmann@informatik.tu-darmstadt.de
[2] Institut National des Sciences Appliquées, Rennes, Rennes, France
domidese@gmail.com

Abstract. Considered is a 2D cellular automaton with moving agents. The objective is to find agents controlled by a Finite State Program (FSP) that can form domino patterns. The quality of a formed pattern is measured by the degree of order computed by counting matching 3×3 patterns (templates). The class of domino patterns is defined by four templates. An agent reacts on its own color, the color in front, and whether it is blocked or not. It can change the color, move or not, and turn into any direction. Four FSP were evolved for multi-agent systems with 1, 2, 4 agents initially placed in the corners of the field. For a 12×12 training field the aimed pattern could be formed with a 100% degree of order. The performance was also high with other field sizes. Livelocks are avoided by using three different variants of the evolved FSP. The degree of order usually fluctuates after reaching a certain threshold, but it can also be stable, and the agents may show the termination by running in a cycle, or by stopping their activity.

Keywords: Cellular automata agents · Multi-agent system · Pattern formation · Evolving FSM behavior · Spatial computing

1 Introduction

Pattern formation is an area of active research in various domains as in physics, chemistry, biology, computer science or natural and artificial life. There exists a lot of examples, namely in polymer composites, laser trapping, spin systems, self-organization, growth processes, morphogenesis, excitable media and so forth [1–10]. Cellular automata (CA) make suitable and powerful tools for catching the influence of the microscopic scale onto the macroscopic behavior of such complex systems [11–13]. At the least, the 1–dimensional Wolfram's "Elementary" CA can be viewed as generating a large diversity of 2–dimensional patterns whenever the time evolution axis is considered as the vertical spatial axis, with patterns depending or not on the random initial configuration [14]. A similar evolution process is observed in the Yamins–Nagpal "1D spatial computer" generating the roughly radial striped pattern of the *Drosophila melanogaster* [15,16]. But the authors emphasize therein how the local-to-global CA paradigm can turn into

V. Malyshkin (Ed.): PaCT 2017, LNCS 10421, pp. 18–31, 2017.
DOI: 10.1007/978-3-319-62932-2_2

the inverse global-to-local question, namely "given a pattern, which agent rules will robustly produce it?"

Based upon our experience from previous works dealing with CA agents and with FSM–agents driven by Finite State Machines and generating spatial patterns [17–19], we focus herein on the problem of generating an optimal configuration of *domino* patterns in an $n \times n$ field, from four 3×3 Moore-neighborhood domino templates. "Optimal" means that the configuration must have neither gap nor overlap. Although the objective in [19] was to form long orthogonal *line* patterns, some similarity will be observed between both configurations as related to alignments in spin systems. A more down-to-earth application for the domino pattern is the problem of packing encountered in different logistics settings, such as the loading of boxes on pallets, the arrangements of pallets in trucks, or cargo stowage [20]. Another application is the construction of a sieve for rectangular particles with a maximum flow rate.

Related Work. (i) *Pattern formation.* A programming language is presented in [15] for pattern-formation of locally-interacting, identical agents – as an example, the layout of a CMOS inverter is formed by agents. Agent-based pattern formations in nature and physics are studied in [21,22]. In [23] a general framework is proposed to discover rules that produce special spatial patterns based on a combination of machine learning strategies including genetic algorithms and artificial neural networks.

(ii) *FSM–controlled agents.* We have designed evolved FSM–controlled CA agents for several tasks, like the *Creature's Exploration Problem* [24,25], the *All-to-All Communication Task* [25–27], the *Target Searching Task* [28], the *Routing Task* [29,30]. The FSM for these tasks were evolved by genetic algorithms mainly. Other related works are a multi-agent system modeled in CA for image processing [31] and modeling the agent's behavior by an FSM with a restricted number of states [32]. An important pioneering work about FSM–controlled agents is [33] and FSM–controlled robots are also well known [34].

This work extends the issues presented in [17–19] with a different class of patterns herein and unlike in [19] only two colors and neither markers nor additional communication signals are used. Furthermore, agents are now able to find patterns with the maximum degree of order. In Sect. 2 the class of target patterns is defined and in Sect. 3 the multi-agent system is presented. In Sect. 4 livelock situations and the termination problem are described. The used genetic algorithm is explained in Sect. 5 and the effectiveness and efficiency of selected FSP are evaluated in Sect. 6 before Conclusion. The CA agents used herein are implemented from the *write* access CA–w concept [35–37].

2 Domino Patterns and Degree of Order

Given a square array of $(n+2) \times (n+2)$ cells including border, we focus on the problem of generating an optimal configuration of *domino* patterns in the $n \times n$ enclosed field, from four domino templates (Fig. 1). The role of the border, with a perimeter of $4n + 4$ white cells, is to facilitate the work of the agents, thus

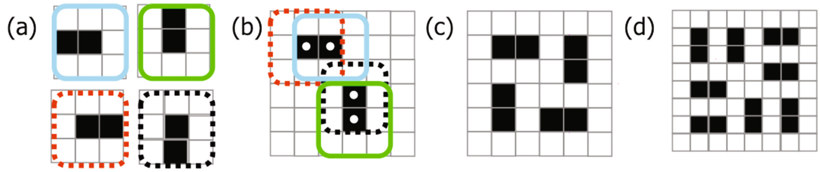

Fig. 1. (a) The four 3×3 domino templates define the domino pattern class. (b) A pattern with 4 hits. It can be tiled (with overlaps) by matching templates. Each matching template produces a hit (dot) in the center. (c) A pattern for a 4×4 field (plus border) with the maximal degree or order 8. (d) A pattern for a 6×6 field with the maximal degree or order 16.

moving within an uniform field. The four possible 3×3 Moore-neighborhood domino templates around a central black cell are displayed in Fig. 1a, showing our so-called spin-like *left* (←), *up* (↑), *right* (→), *down* (↓) dominos. They define the domino pattern *class*.

The templates are tested on each of the n^2 sites (i_x, i_y) of the $n \times n$ field. So each template is applied in parallel on each cell, which can be seen as a classical CA rule application. If a template fits on a site, then a hit (at most one) is stored at this site. Then the sum of all hits is computed which defines the *degree of order* h. A pattern with 4 hits is displayed in Fig. 1b: the top-left horizontal domino is generated by matching the *right* template centered at $(0, 0)$ with the *left* template centered at $(1, 0)$ then producing two hits. In the same way the bottom-right vertical domino is generated by matching the *down* template with the *up* template, thus giving altogether a pattern with order $h = 4$. Dominos are isolated in the sense that neither contact nor overlap is allowed; in other words, a black domino must be surrounded by ten white cells.

Domino Enumeration. For an even side length n, let h_{max} be the maximum expected order. Hereafter we give an evaluation of this optimal order by induction in a non formal way. In (c) and (d) two optimal patterns are displayed respectively for a 4×4 field and a 6×6 field. They are redisplayed in Fig. 2,

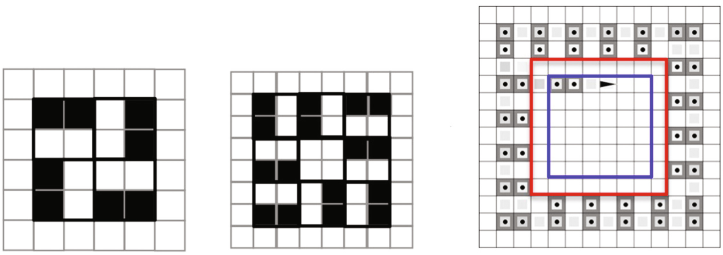

Fig. 2. From left to right: 1. Tiling the 4×4 field with 4 tetraminos. 2. Tiling the 6×6 field with 9 tetraminos. 3. The agent entering the central 6×6 subfield, *with* border, in the 12×12 field: fifth snapshot of Fig. 5a.

showing the patterns now tiled with square 2×2 tetraminos. Such a 4–mino may either contain a domino or be empty. So, a $n \times n$ field (n *even*) can be tiled by exactly $\xi_n^* = n^2/4$ tetraminos. That gives an upper bound for the maximal order. Note that the central 4–mino in the 6×6 field is empty.

Let us now observe the 12×12 field in Fig. 2 showing one agent generating the pattern. Starting from the top-left corner, the agent generates 4 rows of 5 aligned dominos, moving clockwise, before entering a central 6×6 subfield, *with* border. We are now ready for the induction.

Let us call a "*void*" a cell belonging to an *inner* border and let ν_n be the void index in a $n \times n$ field; we claim that $\nu_0 = \nu_2 = \nu_4 = 0$ and

$$\nu_n = 4(n-5) + \nu_{n-6} \qquad (n > 4) \tag{1}$$

and give an informal proof. The first term of the sum is the perimeter (in number of cells) of the inner border surrounding the central $(n-6) \times (n-6)$ subfield, the second term denotes the number of voids in that subfield. Setting $m = n/2$ and $p = \lfloor m/3 \rfloor$ we get

$$\nu_n = \begin{cases} 4p\ (3p-2) & (m \equiv 0) \\ 4p\ (3p) & (m \equiv 1) \\ 4p\ (3p+2) & (m \equiv 2) \end{cases} \pmod 3. \tag{2}$$

The number ξ_n of non-empty 4–minos and bounded by ξ_n^* is then given by

$$\xi_n = \frac{n^2 - \nu_n}{4} \tag{3}$$

Table 1. Domino enumeration for $n \times n$ fields: upper bound ξ_n^*, void index ν_n, domino number ξ_n, optimal degree h_{max}.

n	m	p	ξ_n^*	ν_n	ξ_n	h_{max}
0	0	0	0	0	0	0
2	1	0	1	0	1	2
4	2	0	4	0	4	8
6	3	1	9	4	8	16
8	4	1	16	12	13	26
10	5	1	25	20	20	40
12	6	2	36	32	28	56
14	7	2	49	48	37	74
16	8	2	64	64	48	96
18	9	3	81	84	60	120
20	10	3	100	108	73	146
22	11	3	121	132	88	176
24	12	4	144	160	104	208

namely the domino number, and therefore the maximum expected order is $h_{max} = 2\xi_n$ and the relative order is $h_{rel} = h/h_{max}$. $\square$

The quantities for the first even values of the field size n are displayed in Table 1.

3 Modeling the Multi-agent-System

Compared to classical CA, moving agents with a certain "intelligence" have to be modeled. Therefore the cell rule becomes more complex. Different situations have to be taken into account, such as an agent is situated on a certain cell and is actively performing actions, or an agent is blocked by another agent or by a border cell in front. The *cell state* is modeled as a record of several data items:

$$
\begin{aligned}
&CellState = (Color, Agent)\\
&\quad Color\ L \in \{0, 1\}\\
&\quad Agent = (Activity, Identifier, Direction, ControlState)\\
&\qquad\quad Activity \in \{\texttt{true}, \texttt{false}\}\\
&\qquad\quad Identifier\ ID \in \{0, 1, ..., k-1\}\\
&\qquad\quad Direction\ D \in \{0, 1, 2, 3\} \equiv \{\texttt{toN},\ \texttt{toE},\ \texttt{toS},\ \texttt{toW}\}\\
&\qquad\quad ControlState\ S \in \{0, 1, ..., N_{states} - 1\}.
\end{aligned}
$$

This means that each cell contains a potential agent, which is either active and visible or passive and not visible. When an agent is moving from A to B, its whole state is copied from A to B and the *Activity* bit of A is set to false. The agent's structure is depicted in Fig. 3. The finite state machine (FSM) realizes the "brain" or control unit of the agent. Embedded in the FSM is a state table which defines the actual behavior. The state table can also be seen as a program or algorithm. Therefore the abbreviations FSP (*finite state program*) or AA (*agent's algorithm*) are preferred herein. Outputs are the actions and the next control state. Inputs are the control state s and defined input situations x.

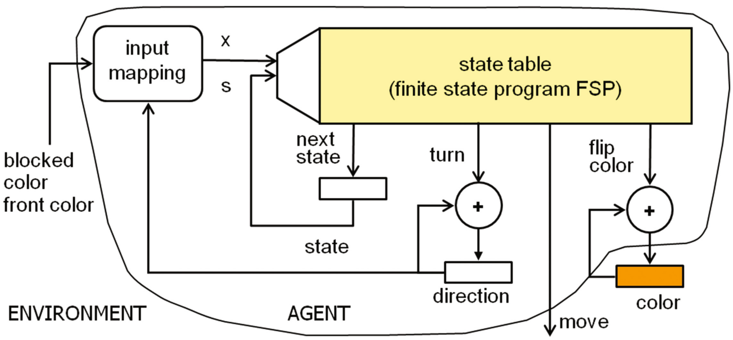

Fig. 3. An agent is controlled by a finite state machine (FSM). The state table defines the agent's next control state, its next direction, and whether to move or not. The table also defines whether the color shall be toggled ($0 \rightarrow 1$) or ($1 \rightarrow 0$).

An input mapping function is used in order to limit the size of the state table. The *input mapping* reduces all possible input combinations to an index $x \in X = \{0, 1, \ldots, N_x - 1\}$ used in combination with the control state to select the actual line of the state table.

The capabilities of the agents have to be defined before designing or searching for an AA. The main capabilities are: the perceivable inputs from the environment, the outputs and actions an agent can perform, the capacity of its memory (number of possible control and data states) and its "intelligence" (useful proactive and reactive activity). Here the intelligence is limited and carried out by a mapping of its state and inputs to the next state, actions and outputs.

An agent can react on the following inputs:

- **control state:** agent's control state s,
- **direction:** agent's direction D,
- **color:** color L of the cell the agent is situated on,
- **front color:** color L_F of the cell in front,
- **blocked:** the blocking condition caused either by a border, another agent in front, or in case of a conflict when another agent gets priority to move to the front cell. The inverse condition is called *free*.

An agent can perform the following actions:

- **next state:** $state \leftarrow nextstate \in \{0, ..., N_{states} - 1\}$.
- **move:** $move \in \{0, 1\} \equiv \{wait, go\}$.
- **turn:** $turn \in \{0, 1, 2, 3\}$.
  The new direction is $D(t + 1) \leftarrow (D(t) + turn)\ mod\ 4$.
- **flip color:** $flipcolor \in \{0, 1\}$.
  The new color is $L(t + 1) \leftarrow (L(t) + flipcolor)\ mod\ 2$.

An agent has a moving direction D that also selects the cell in front as the actual neighbor. What can an agent observe from a neighboring cell? In our model it can only detect the blocking condition and the color in front. So the agents' sensing capabilities are weak.

All actions can be performed in parallel. There is only one constraint: when the agent's action is *go* and the situation is *blocked*, then an agent cannot move and has to wait, but still it can turn and change the cell's color. In case of a moving conflict, the agent with the lowest identifier (ID $= 0...k-1$) gets priority. Instead of using the identifier for prioritization, it would be possible to use other schemes, e.g. random priority, or a cyclic priority with a fixed or space-dependent base. The following input mapping was used, $x \in \{0, 1, \ldots, 7\}$:

$x = 0 + 4b$, if $color = 0$ and $frontcolor = 0$
$x = 1 + 4b$, if $color = 1$ and $frontcolor = 1$
$x = 2 + 4b$, if $color = 0$ and $frontcolor = 1$
$x = 3 + 4b$, if $color = 1$ and $frontcolor = 0$

where $b = 0$ if *free*, otherwise $b = 1$ if *blocked*. This mapping was designed by experience from former work. Of course, other input mappings are possible, with more or less x codes, or other assignments, e.g. more neighbors could be taken into account, the blocking conditions could be distinguished (by border, by agent, by conflict), or a part of the agent's private control state could be presented to the neighbors. Note that the sensing capabilities are quite limited, and that makes the given task difficult to solve.

4 Livelock and Termination

The Livelock Problem. During the work of evolving FSP, it turned out that livelocks may appear for systems with more than one agent. In a livelock the agents act in a way that there is no more progress in the system's global state towards the aimed pattern. An analogy is when two people meet head-on and each tries to step around the other, but they end up swaying from side to side, getting in each other way as they try to get out of the way. Here livelocks appeared when 2 or 4 agents were placed symmetrically in space. Then the state/actions sequence was the same and the agents got stuck in cyclic paths. Fortunately we found a simple way to avoid them. Three variants of an FSP are used. Agents start in three different control states, depending on the agent's identifier: $initial\ state = ID\ mod\ 3$. By this technique we were able to find FSP without livelocks, agents can now show three different behaviors. As we cannot influence the structure of the evolved FSP, the FSP state's graph may have different prefix state sequences, or the FSP may even fall into three separate graphs (co-evolution of up to three FSP). This means that the genetic algorithm automatically finds the best choice of more equal or more distinct FSP under the restriction of a given maximal number of states N_{states}.

The Termination Problem. How can the multi-agent system be stopped in a decentralized way after having reached the required degree of order? One idea is to communicate the hits all-to-all. Thereby the difficulty is that pattern and degree of order are usually changing over time, and the transportation of the hit information is delayed in space. So it would be more elegant, if the system state (pattern or hit-count) reaches automatically a fixed point. We define for our multi-agent system that has reached a certain degree of order

(1) *Soft-termination:* The pattern is stable, and there exists one agent that is active (moves and/or changes direction).
(2) *Hard-termination:* The pattern is stable, and all agents are passive (not moving and/or not changing direction).

The termination problem has been studied for distributed systems, and now it is under research also for multi-agent systems [38].

5 Evolving FSP by a Genetic Algorithm

An ultimate aim could be to find an FSP that is optimal for all possible initial configurations on average. This aim is very difficult to reach because it needs a huge amount of computation time. Furthermore, it depends on the question whether all-rounders or specialists are favored. Therefore, in this work we searched only for *specialists* optimized for (i) a fixed field size of $N = n \times n$, $n = 12$, (ii) 4 special initial configurations with 1, 2, 4 agents where the agents are placed in the corners of the field. The number of different FSP which can be coded by a state table is $Z = (|s||y|)^{(|s||x|)}$ where $|s|$ is the number of control states, $|x|$ is the number of inputs and $|y|$ is the number of outputs. As the search space increases exponentially, we restricted the number of inputs to $|x| = 8$ and the number of states to $|s| = N_{states} = 18$. Experiments with lower numbers of states did not yield the aimed quality of solutions.

A relatively simple genetic algorithm similar to the one in [17] was used in order to find (sub)optimal FSP with reasonable computational cost. A possible FSP solution corresponds to the contents of the FSM's state table. For each input combination `(x, state) = j`, a list of actions is assigned:

```
actions(j) = (nextstate(j), move(j), turn(j), flipcolor(j))
```

as displayed on the FSP genome in Fig. 4.

The fitness is defined as the number t of time steps which is necessary to emerge successfully a target pattern with a given degree h_{target} of order, averaged over all given initial random configurations. "Successfully" means that a target pattern with $h \geq h_{target}$ was found. The fitness function F is evaluated by simulating the system with a tentative FSP_i on a given initial configuration. Then the mean fitness $\overline{F}(FSP_i)$ is computed by averaging over all initial configurations of the training set. $\overline{F}$ is then used to rank and sort the FSP.

In general it turned out that it was very time consuming to find good solutions with a high degree of order, due to the difficulty of the agents' task in relation to their capabilities. Furthermore the search space is very large and difficult to explore. The total computation time on a Intel Xeon QuadCore 2 GHz was around 4 weeks to find all needed FSP.

Evolved Finite State Programs. The used fields are of size $N = n \times n$. The cell index (i_x, i_y) starts from the top left corner $(0, 0)$ to the bottom right corner $(n-1, n-1)$. The top right corner is $(n-1, 0)$. The index K defines a set of initial configurations. Here only 4 initial configurations are used:

$K = 1$: 1 agent with direction $\rightarrow$, placed at $(0, 0)$

$K = 2$: 2 agents, one placed like in configuration $K = 1$, and another with direction $\leftarrow$ placed at $(n-1, n-1)$

$K = 4$: 4 agents, two of them placed like in configuration $K = 2$, and another with direction $\downarrow$ placed at $(n-1, 0)$, and another with direction $\uparrow$ placed at $(0, n-1)$

$K = 124$: This index specifies a set of configurations, the union of $K = 1$, $K = 2$, and $K = 4$

```
     state  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
           /x=0                                                \   /x=1                                               \
 nextstate 10  4 15  1  2 10 10  1  1 12  7  6 12  9  5 11  3  2    5  1  3 11  2  5  7  6 10 13  7  8  6  3  1 17 11 10
 flipcolor  0  1  0  0  1  0  1  1  0  0  0  0  0  0  1  0  0  0    0  0  0  0  0  0  0  1  1  1  1  0  0  1  1  0  1  0
      move  0  1  0  0  0  0  0  1  0  0  1  0  1  0  0  1  0  1    0  0  0  0  0  1  1  0  1  1  0  1  1  0  0  0  0  0
      turn  1  1  2  1  3  1  2  1  3  3  2  3  3  2  2  2  2  0    0  1  2  3  1  3  0  1  1  2  3  0  2  1  0  0  1  2
           /x=2                                                \   /x=3                                               \
 nextstate  3  6 10 16  2 13  9  7  9  5  4  0 17 15 17  4 15 17    1  5  5  6  5  4  1 13 17  8  0 11 17  2 12  5  4  7
 flipcolor  1  0  0  0  1  0  1  1  1  1  0  0  1  0  0  1  0  1    0  0  1  1  0  1  0  0  1  1  1  1  0  0  0  0  0  1
      move  1  1  1  1  0  1  1  1  0  1  1  1  1  0  1  0  0  0    1  0  1  1  1  1  1  1  0  1  1  1  0  0  1  1  0  1
      turn  2  3  2  0  2  1  2  0  1  1  1  3  3  3  0  1  1  3    0  1  0  2  0  3  0  1  3  2  3  1  3  1  3  1  0  2
           /x=4                                                \   /x=5                                               \
 nextstate  4 16 12 16 15 10  8  1  1  5  7  1  6  0 15 17 17  6   10  1 12 16 17 16  7 10  5  7  8  6 14  3 12  3  0  6
 flipcolor  1  0  1  1  0  1  1  0  1  1  1  0  1  1  1  1  1  0    1  1  0  1  0  0  0  1  1  0  0  1  1  0  0  0  0  0
      move  1  0  1  0  0  0  1  1  1  1  1  1  0  0  0  1  1  0    1  1  0  0  0  1  1  1  0  1  1  0  1  0  0  1  0  0
      turn  0  1  1  1  2  0  3  0  1  2  1  2  1  2  1  3  0  0    1  0  1  3  0  2  0  2  1  1  1  1  1  2  2  3  2  0
           /x=6                                                \   /x=7                                               \
 nextstate  0 12  7  2  7 11  0 15  0  3  5  2  0  2 10  7  5 16   17 17  9  2 11  5  6  2  0 10  2  8 10  4  6  5  4  8
 flipcolor  1  0  0  1  1  0  0  0  0  0  1  1  0  0  1  0  0  1    0  1  1  1  1  1  1  0  0  1  0  0  1  1  0  1  0  0
      move  1  0  0  1  1  1  1  0  0  1  1  1  1  0  1  0  0  1    1  0  0  0  1  0  0  1  0  0  1  1  1  0  0  1  1  0
      turn  1  2  2  0  2  2  3  1  2  1  1  0  0  3  3  2  2  1    0  2  0  1  1  2  0  3  2  2  2  1  2  1  3  1  1  0
```

Fig. 4. FSP genome with $N_{states} = 18$ states and $N_x = 8$ inputs.

The best found FSP is denoted by

FSP^{n,K,h}**:** for field size n, configuration K, and a reached order h_{max}. The reached order can also be given relatively as h_{rel} with percent suffix

The following four FSP were evolved by the genetic algorithm:

$$FSP^{12,1,100\%} = FSP^{12,1,56}$$
$$FSP^{12,2,100\%} = FSP^{12,2,56}$$
$$FSP^{12,4,100\%} = FSP^{12,4,56}$$

and the more general mixed one $FSP^{12,124,100\%}$ that is 100% successful on each of the 3 initial fields for $K = 1, 2, 4$. Its genome is displayed in Fig. 4. Note that $h_{max} = 56$ for $n = 12$ according to Table 1 whence $h_{rel} = 100\%$.

6 Simulation and Performance Evaluation

Simulation. Firstly, the agent-system was simulated and observed for the first three evolved programs ($FSP^{12,1,56}$, $FSP^{12,2,56}$, $FSP^{12,4,56}$). Figure 5 shows the time evolution of the domino pattern for the system with 1, 2, and 4 agents. The strategy of 1 agent is to move along the border clockwise (Fig. 5a) and then after one cycle moving inwards. Roughly the path is close to a spiral. Looking to the path in detail, the agent moves more or less back and forth in order to build the optimal pattern. Thereby already built dominoes can be destructed and rebuilt in a different way. An optimal pattern with $h_{max} = 56$ is built at $t = 215$.

The systems with two agents (Fig. 5b) and four agents (Fig. 5c) follow a similar strategy, but the work is shared and each agent cooperates in building the optimal pattern. The cooperation is achieved by detecting dominoes already in place and then rearrange them in a better way or move just inwards to the empty area in order to create new dominoes. The optimal pattern is built at $t = 154$ for the 2-agent-system, and at $t = 51$ for the 4-agent-system.

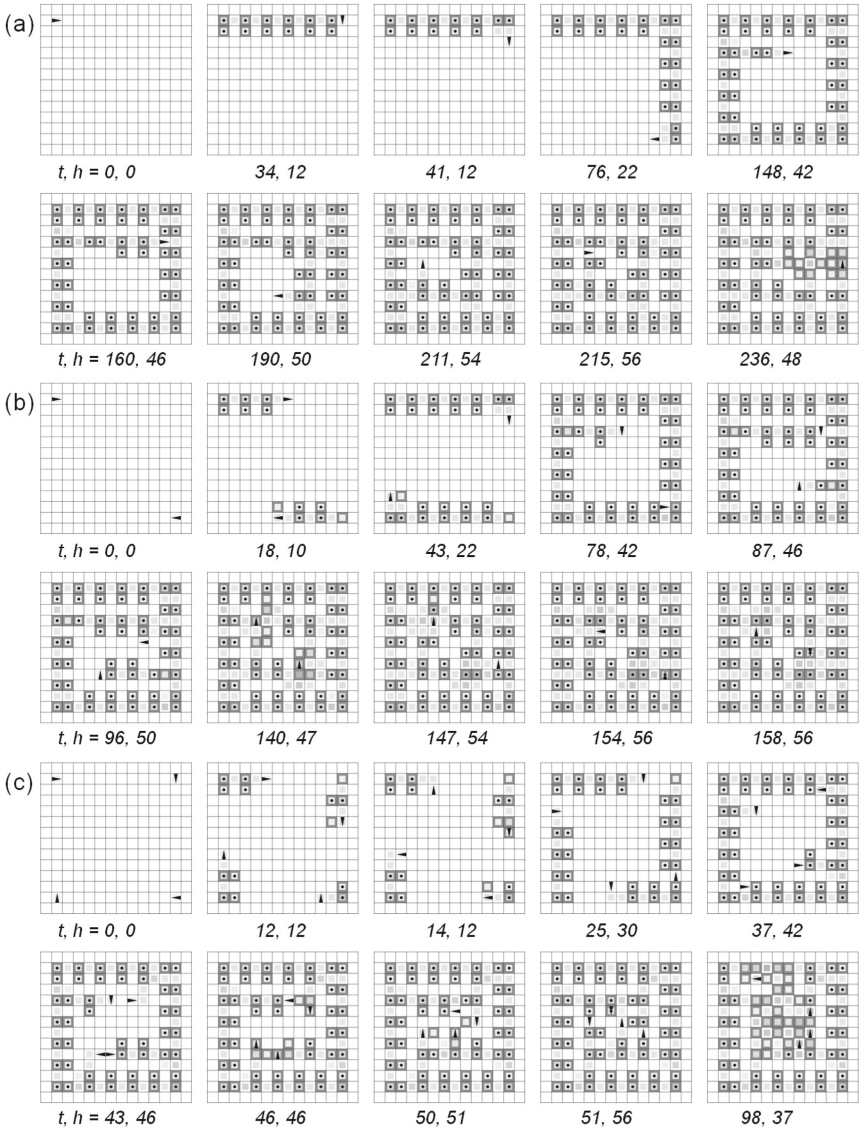

Fig. 5. Dots are marking the hits. Inner squares in light grey are marking visited cells, the darker the more often visited. (a) 1-agent-system. The agent starts in the left corner and moves mainly clockwise, and from the border to the centre. At $t = 215$ an optimal pattern with $h = 56$ is formed. For $t \geq 236$ the pattern remains stable with $h = 48$. (b) 2-agent-system. The agents are building the pattern together. Agent 0 and 1 use a slightly different algorithm, see configuration at $t = 18$. For $t \geq 158$ the agents run in a cycle without changing the optimal pattern. (c) 4-agent-system. The agents 0, 1, 2 use slightly different algorithms, see configuration at $t = 12$. At $(t = 98, h = 37)$ all agents have stopped their activities.

Termination. What happens after having built the optimal pattern?

1-agent-system: During $t = 215 \ldots 235$ the agent continues its walk in the direction of the right border, thereby changing the pattern's order in the sequence $h = 56, 52, 56, 54, 56, 54, 52, 50, 48$. Then, for $t \geq 236$ the pattern remains stable with $h = 48$, and then for $t \geq 240$ the agent is running a 4-step-cycle within a block of 2×2 cells. So we have a *non-optimal soft-termination* with $h_{rel} = 48/56$.

2-agent-system: For $t = 154, 155, 156, 157, 158^{+}$ the agents change slightly the order to $h = 56, 52, 52, 52, 56$. Then for $t \geq 158$ each of them runs in a cycle of period 8 following a square path within a block of 3×3 cells without changing the optimal pattern. This means an *optimal soft-termination* with $h_{rel} = 100\%$. This result was not expected and was not explicitly forced by the genetic. But it shows that optimal terminations in such multi-agent-systems are possible and can be evolved.

4-agent-system: For $t = 51, 52, \ldots 98^{+}$ the agents are reducing the order to $h = 56, 50, \ldots 37$ with fluctuations. At $(t = 60, h = 49)$ agent 1 stops its activities, and then at $(t = 63, h = 47)$ agent 0 stops its activities, and then at $(t = 75, h = 44)$ agent 3 stops its activities, and then at $(t = 98, h = 37)$ agent 2 stops its activities. For $t \geq 98$ all agents have stopped their activities, this means a *non-optimal hard-termination* (see last snapshot in Fig. 5c). Agent with $ID = 0, 1, 2, 3$ started at position $(0, 0), (n-1, 0), (n-1, n-1), (0, n-1)$ respectively. Agents 0 and 3 are using the same variant of the FSP whereas agents 1 and 2 use other variants.

Table 2. Performance of the k–agent systems, especially evolved for each k. The 4–agent system is almost 4 times faster than the 1–agent system.

Program	Agents k	Time t_k	Time per cell t_k/N	Speedup $S = t_1/t_k$	Efficiency S/k
$FSP^{12,1,100\%}$	1	215	1.49	1.00	1.00
$FSP^{12,2,100\%}$	2	154	1.07	1.40	0.70
$FSP^{12,4,100\%}$	4	54	0.38	3.98	0.99

Comparison with the More General, Mixed $\mathbf{FSP^{12,124,100\%}}$. The mixed FSP was evolved to work with 1, 2, or 4 agents, therefore it is more general. Now the time to reach 100% success is longer, $t_{124} = 255, 180, 105$ for $k = 1, 2, 4$. Compared to the optimal time of the special FSP presented before and given in Table 2 the ratio is $t_{124}/t(k) = 1.19, 1.17, 1.94$. That means that special evolved algorithms may save significant computation time, in our example up to 94%.

Performance of the Mixed FSP for Other Field Sizes. Now it was tested how sensitive the mixed FSP is against a change of the field size. It was required that all k–agent systems $(k = 1, 2, 4)$ are successful to the up most reachable degree h_{rel}^{max}. It was found by incrementing h_{rel} to the point where at least one agent-system was not successful. Table 3 shows the times t_k to order the systems up to h_{rel}^{max} (refer to Eq. 3 and Table 1 for h_{max}). In order to compare

Table 3. Used was the mixed FSP evolved for field size 12×12. The times t_k for different field sizes was recorded for the maximal reachable degree of order.

Field size	4×4	6×6	8×8	10×10	12×12	14×14	16×16
Reached h_{rel}^{max}	$4/8 = 50\%$	$14/16 = 88\%$	$22/26 = 85\%$	$34/40 = 85\%$	$56/56 = 100\%$	$72/74 = 97\%$	$88/96 = 92\%$
t_1	10	52	82	125	255	303	351
t_2	13	30	42	62	180	216	272
t_4	4	19	21	32	105	123	161

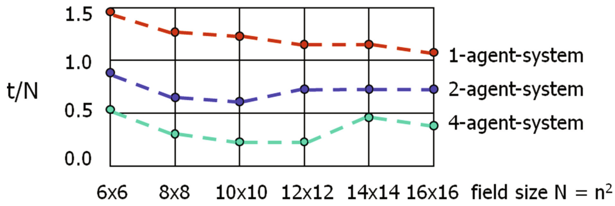

Fig. 6. Time steps per cell needed to order the fields with a degree $h_{rel} \geq 85\%$.

the performance for different field sizes the metric t/N (time steps per cell) was used. Furthermore a fixed bound for $h_{rel} = 85\%$ was used. Then the time was measured for this bound. The outcome is depicted in Fig. 6. The normalized time t/N is minimal at $n = 16, 10, 10$ for $k = 1, 2, 4$ and the 4-agent system is around 3 times faster than the 1-agent-system.

7 Conclusion

The class of the aimed domino patterns was defined by four templates (3×3 local patterns). Four FSP were evolved for multi-agent systems with 1, 2, 4 agents initially placed in the corners of the field. The reached degree of order was 100% for the 12×12 training field, and greater than 85% for field sizes between 6×6 and 16×16. Livelocks were avoided by using up to three different variants of the FSP depending on the agent's identifier. These variants use different initial control states and may show a totally different individual behavior. This can be interpreted as a co-evolution of three cooperating behaviors. It was observed that the achieved pattern can reach a stable fixed point, and then the agents run in small cycles or even stop their activities totally. Further work is directed to the termination problem, the co-evolution, and the problem of finding robust multi-agent systems that can order fields of any size perfectly.

References

1. Shi, D., He, P., Lian, J., Chaud, X., Bud'ko, S.L., Beaugnon, E., Wang, L.M., Ewing, R.C., Tournier, R.: Magnetic alignment of carbon nanofibers in polymer composites and anisotropy of mechanical properties. J. App. Phys. **97**, 064312 (2005)
2. Itoh, M., Takahira, M., Yatagai, T.: Spatial arrangement of small particles by imaging laser trapping system. Opt. Rev. **5**(1), 55–58 (1998)
3. Jiang, Y., Narushima, T., Okamoto, H.: Nonlinear optical effects in trapping nanoparticles with femtosecond pulses. Nat. Phys. **6**, 1005–1009 (2010)
4. Niss, M.: History of the Lenz-Ising model, 1920–1950: from ferromagnetic to cooperative phenomena. Arch. Hist. Exact Sci. **59**(3), 267–318 (2005)
5. Press, D., Ladd, T.D., Zhang, B., Yamamoto, Y.: Complete quantum control of a single quantum dot spin using ultrafast optical pulses. Nature **456**, 218–221 (2008)
6. Bagnold, R.E.: The Physics of Blown Sand and Desert Dunes. Chapmann and Hall, Methuen, London (1941)
7. Turing, A.M.: The chemical basis of morphogenesis. Philos. Trans. R. Soc. Lond. **B237**, 37–72 (1952)
8. Tyson, J.J.: The Belousov-Zhabotinskii Reaction. Lecture Notes in Biomathematics. Springer, Heidelberg (1976). doi:10.1007/978-3-642-93046-1
9. Greenberg, J.M., Hastings, S.P.: Spatial patterns for discrete models of diffusion in excitable media. SIAM J. Appl. Math. **34**(3), 515–523 (1978)
10. Progogine, I., Stengers, I.: Order out of Chaos. Heinemann, London (1983)
11. Chopard, B., Droz, M.: Cellular Automata Modeling of Physical Systems. Cambridge University Press, Cambridge (1998)
12. Deutsch, A., Dormann, S.: Cellular Automaton Modeling of Biological Pattern Formation. Birkäuser, Boston (2005)
13. Désérable, D., Dupont, P., Hellou, M., Kamali-Bernard, S.: Cellular automata in complex matter. Complex Syst. **20**(1), 67–91 (2011)
14. Wolfram, S.: Statistical mechanics of cellular automata. Rev. Mod. Phys. **55**(3), 601–644 (1983)
15. Nagpal, R.: Programmable pattern-formation and scale-independence. In: Minai, A.A., Bar-Yam, Y. (eds.) Unifying Themes in Complex Sytems IV, pp. 275–282. Springer, Heidelberg (2008). doi:10.1007/978-3-540-73849-7_31
16. Yamins, D., Nagpal, R., Automated global-to-local programming in 1-D spatial multi-agent systems. In: Proceedings of the 7th International Conference on AAMAS, pp. 615–622 (2008)
17. Hoffmann, R.: How agents can form a specific pattern. In: Wąs, J., Sirakoulis, G.C., Bandini, S. (eds.) ACRI 2014. LNCS, vol. 8751, pp. 660–669. Springer, Cham (2014). doi:10.1007/978-3-319-11520-7_70
18. Hoffmann, R.: Cellular automata agents form path patterns effectively. Acta Phys. Pol. B Proc. Suppl. **9**(1), 63–75 (2016)
19. Hoffmann, R., Désérable, D.: Line patterns formed by cellular automata agents. In: El Yacoubi, S., Wąs, J., Bandini, S. (eds.) ACRI 2016. LNCS, vol. 9863, pp. 424–434. Springer, Cham (2016). doi:10.1007/978-3-319-44365-2_42
20. Birgin, E.G., Lobato, R.D., Morabito, R.: An effective recursive partitioning approach for the packing of identical rectangles in a rectangle. J. Oper. Res. Soc. **61**, 303–320 (2010)
21. Bonabeau, E.: From classical models of morphogenesis to agent-based models of pattern formation. Artif. Life **3**(3), 191–211 (1997)

22. Hamann, H., Schmickl, T., Crailsheim, K.: Self-organized pattern formation in a swarm system as a transient phenomenon of non-linear dynamics. Math. Comput. Mod. Dyn. Syst. **18**(1), 39–50 (2012)
23. Bandini, S., Vanneschi, L., Wuensche, A., Shehata, A.B.: A neuro-genetic framework for pattern recognition in complex systems. Fundam. Inf. **87**(2), 207–226 (2008)
24. Halbach, M., Hoffmann, R., Both, L.: Optimal 6-state algorithms for the behavior of several moving creatures. In: Yacoubi, S., Chopard, B., Bandini, S. (eds.) ACRI 2006. LNCS, vol. 4173, pp. 571–581. Springer, Heidelberg (2006). doi:10.1007/11861201_66
25. Ediger, P., Hoffmann, R.: Optimizing the creature's rule for all-to-all communication. In: Adamatzky, A., et al. (eds.) Automata 2008, pp. 398–412 (2008)
26. Ediger, P., Hoffmann, R.: Solving all-to-all communication with CA agents more effectively with flags. In: Malyshkin, V. (ed.) PaCT 2009. LNCS, vol. 5698, pp. 182–193. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03275-2_19
27. Hoffmann, R., Désérable, D.: All-to-all communication with cellular automata agents in 2D grids. J. Supercomput. **69**(1), 70–80 (2014)
28. Ediger, P., Hoffmann, R.: CA models for target searching agents. Elec. Notes Theor. Comput. Sci. **252**, 41–54 (2009)
29. Ediger, P., Hoffmann, R., Désérable, D.: Routing in the triangular grid with evolved agents. J. Cell. Autom. **7**(1), 47–65 (2012)
30. Ediger, P., Hoffmann, R., Désérable, D.: Rectangular vs triangular routing with evolved agents. J. Cell. Autom. **8**(1–2), 73–89 (2013)
31. Komann, M., Mainka, A., Fey, D.: Comparison of evolving uniform, non-uniform cellular automaton, and genetic programming for centroid detection with hardware agents. In: Malyshkin, V. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 432–441. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73940-1_43
32. Mesot, B., Sanchez, E., Peña, C.-A., Perez-Uribe, A.: SOS++: finding smart behaviors using learning and evolution. In: Artificial Life VIII, pp. 264–273. MIT Press (2002)
33. Blum, M., Sakoda, W.J.: On the capability of finite automata in 2 and 3 dimensional space. In: SFCS 1977, pp. 147–161 (1977)
34. Rosenberg, A.L.: Algorithmic insights into finite-state robots. In: Sirakoulis, G., Adamatzky, A. (eds.) Robots and Lattice Automata. Emergence, Complexity and Computation, vol. 13, pp. 1–31. Springer, Cham (2015). doi:10.1007/978-3-319-10924-4_1
35. Hoffmann, R.: The GCA-w massively parallel model. In: Malyshkin, V. (ed.) PaCT 2009. LNCS, vol. 5698, pp. 194–206. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03275-2_20
36. Hoffmann, R.: Rotor-routing algorithms described by CA-w. Acta Phys. Pol. B Proc. Suppl. **5**(1), 53–67 (2012)
37. Hoffmann, R., Désérable, D.: Routing by cellular automata agents in the triangular lattice. In: Sirakoulis, G., Adamatzky, A. (eds.) Robots and Lattice Automata. Emergence, Complexity and Computation, vol. 13, pp. 117–147. Springer, Cham (2015). doi:10.1007/978-3-319-10924-4_6
38. Lahlouhi, A.: MAS-td: an approach to termination detection of multi-agent systems. In: Gelbukh, A., Espinoza, F.C., Galicia-Haro, S.N. (eds.) MICAI 2014. LNCS, vol. 8856, pp. 472–482. Springer, Cham (2014). doi:10.1007/978-3-319-13647-9_42