Line Patterns Formed 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, France
domidese@gmail.com

Abstract. Considered is a 2D cellular automaton with moving agents. Each cell contains a particle with a certain spin/color that can be turned by an agent. Four colors are used. The objective is to align the spins in parallel along horizontal and vertical lines, in order to form long orthogonal "line patterns". The quality of a line pattern is measured by a degree of order computed by counting matching 3 x 3 patterns. Additional markers are used and signals between agents are introduced in order to improve the task efficiency. The agents' behavior is controlled by a finite state machine (FSM). An agent can perform 128 actions altogether as combinations of moving, turning, color changing, marker setting and signaling. It reacts on its own state and on the sensed colors, markers and signals. For a given set of n x n fields, near optimal FSM were evolved by a genetic algorithm. The evolved agents are capable of forming line patterns with a limited degree of order. The scalability of two FSM against a varying number of agents is studied as well as the efficiency gain through the newly introduced signals.

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

1 Introduction

How can cellular automata agents (CAA) be configured in order to form specific structures? CAA are *agents* modeled by classical CA using relatively complex rules. There are many applications, e.g. the forming of mechanical, chemical, biological [1] or artificial structures, or the building of computational devices and communication networks. Nano-structures can be built by nano-robots, or by beaming focused energy onto certain cells in order to change their physical state [2–6]. Building special nano-polymers or functional polymers could be another application. The alignment of spins – as discussed in this paper – allows to minimize the conductivity of a thin-film structure based on the giant magnetoresistance.

The Task. Given is a field of $N = n \times n$ cells with border, n assumed to be even. A given number k of agents is moving around in the field. The agents' task

S. El Yacoubi et al. (Eds.): ACRI 2016, LNCS 9863, pp. 424–434, 2016.
DOI: 10.1007/978-3-319-44365-2_42

is to construct a global state where a certain spatial *line pattern* appears that belongs to a predefined line pattern class. The agents behavior is controlled by an embedded finite state machine (FSM). The main objective is to find "intelligent" agents that are able to solve this task. Each cell, except the border cells, contains a particle with a certain color/spin, $color \in \{0, 1, 2, 3\}$. The border cells' color is NIL with value -1. In addition, *markers* are available in each cell, $marker \in \{0, 1\}$. An agent can change color and marker on its site. Markers are used as a distributed global memory. A marker can later be used by the same agent or by another agent. Markers enlarge the control and data space and allow indirect communication between agents: e.g. an agent may set a marker from 0 to 1 indicating that the current site is already locally ordered. As an extension to previous works [7,8], explicit *signals* between agents are introduced. Now a further objective is to study how signals act onto system performance. Initially the color, marker and agents' position and direction are randomly distributed.

We define the aimed *line pattern class* by long horizontal lines of parallel up-or-down spins, or vertical lines of right-or-left spins.The neighboring cells of a line shall have another spin/color. The capabilities of the agents shall be constrained, i.e. the number of control states, the action set and the amount of perceived information.

Agents. What is the advantage to solve this task by agents? Generally speaking, agents can behave in a flexible, powerful and coordinated way because of their intelligence and their specific sensors and actuators. Important properties that can be achieved by agents are: (*Scalability*) The problem can usually be solved with a variable number of agents, and faster with more agents in a certain range; (*Tuneability*) The problem can be solved faster or with a higher quality by increasing the agent's intelligence; (*Flexibility*) Similar problems can be solved by the same agents, e.g. when the shape or size of the environment is changed; (*Fault-tolerance*) When obstacles are introduced or some agents are faulty, the problem can still be solved in a gracefully degraded way; (*Updating-tolerance*) Often the time-evoluted global state depends only weakly on the updating-scheme (synchronous, asynchronous). This is important if no global clock is available.

Related Work. (i) *FSM-controlled agents:* We have designed evolved FSM-controlled CAA for several tasks, like the *Creature's Exploration Problem* [9, 10], the *All-to-All Communication Task* [10–12], the *Target Searching Task* [13], the *Routing Task* [14,15]. 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 [16], and modeling the agent's behavior by an FSM with a restricted number of states [17]. An important pioneering work about FSM-controlled agents is [18]. FSM-controlled robots are also well known.

(ii) *Pattern formation:* Agent-based pattern formations in nature and physics are studied in [19,20]. A programming language is presented in [21] for pattern-formation of locally-interacting, identically-programmed agents – as an example, the layout of a CMOS inverter is formed by agents;related is a new "Global–to–Local" theory emerging in [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.

(iii) *Modeling moving agents:* CAA used herein are modeled within the CA paradigm [7] and implemented from the *write access* CA–w concept [24–26] allowing a cell to write information onto a neighbor. Other modeling concepts related to CA are lattice-gas cellular automata [27], block substitutions [28] or partitioned CA as used in [12].

This work extends the issues presented in [7]. The class of patterns was different therein and satisfactory patterns could only be found with two colors. Now the aim is to generate patterns with four colors using explicit markers and additional communication signals. In Sect. 2 the class of target patterns and a measure for them are defined and in Sect. 3 the multi-agent system with its agent's actions, sensing features and control structure is presented. The used genetic algorithm is explained in Sect. 4, the effectiveness and efficiency of selected FSMs and signals are evaluated in Sect. 5.

2 Line Patterns and Degree of Order

How can the class of line patterns be defined? The idea is to use a set of small $m \times m$ matching patterns (or *templates*) and test them on each site (x,y) of the cell 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. For our problem the size of the templates is 3×3. Larger templates could be used if more sophisticated patterns should be generated. The process is illustrated in Fig. 1. In order to detect lines more easily in a pattern, small rectangles with redundant colors (b) are used in the representation instead of the spin-arrows (a). A black dot is used to display a hit (c). The used templates are depicted in (d–g): a (horizontal) row of 3 parallel up-spins (d), a (vertical)

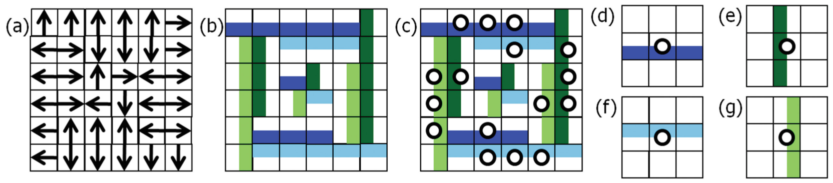

Fig. 1. (a) An optimal balanced line pattern of a 6×6 field with a maximum number of hits. A line is defined by a row or column of parallel spins. (b) Another representation of (a), each spin is represented by a specific rectangle within a square. Additional redundant colors are used in order to make the lines more distinguishably. (c) Hits are depicted as dots. (d–g) The templates defining the class of line patterns. Templates are local test patterns which are expected to appear in a target pattern. (Color figure online)

column of 3 parallel right-spins (e), a row of 3 parallel down-spins (f), a column of 3 parallel left-spins (g). The six empty sites of each template have a color different from the line's color. Thereby two or more parallel lines with the same color/spin are not allowed.

A line of length $w > 2$ has a hit count $h = w - 2$ because the beginning and ending of a line are not counted. This means also that lines of length 1 and 2 are not counted. Not counting the terminal cells of each line can be seen as a penalty reflected in the fitness function which is used during the optimization process, thereby the searching for long lines is favored. The aimed patterns consist in a maximum number of long lines. The theoretically maximum order is at least $h_{max} = (n-2)^2$ (for even side length n). The relative order is $h_{rel} = h/h_{max}$. The maximum can be reached by optimal line patterns, e.g. the one shown in Fig. 1(a–b). Patterns are *balanced* if the frequency for each color is the same, i.e. $\forall i \in \{0, 1, 2, 3\} : n^2/4 = \sum(color(x, y) = i)$ and *unbalanced* otherwise. Unbalanced patterns can be generated easier than balanced patterns and can even have higher hit counts than optimal balanced patterns.

3 Modeling the Multi-Agent-System

The cell rule has to react on several non-uniform situations, such as:an agent is actively operating on that cell, an agent is in the neighborhood, or a border cell is in front. Therefore the cell state is modeled as a record of several data items.

$CellState = (Border, Color, Marker, Agent)$, where
$Border \in \{true, false\}$, $Color\ L \in \{0, 1, 2, 3\}$, $Marker\ M \in \{0, 1\}$
$Agent = (Activity, Identifier, Direction, ControlState)$
$Activity \in \{true, false\}$, $Identifier\ ID \in \{0, 1, ..., k-1\}$
$Direction\ D \in \{0, 1, 2, 3\} \equiv \{toN,\ toE,\ toS,\ toW\}$
$ControlState\ S \in \{0, 1, ..., N_{states} - 1\}$.

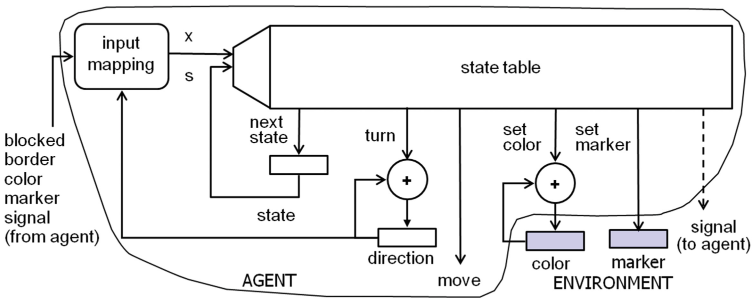

Fig. 2. An agent is controlled by a finite state machine (FSM). The state table defines the agent's next control state, its next direction, and whether to move or not. The table also defines the next color, the next marker and the signal to another agent. (Color figure online)

This means that each cell is equipped with a potential agent, which is either active or not. When an agent is moving from A to B, it is copied from A to B and the *Activity* bit on A is set to false. The agent's behavior is depicted in Fig. 2. The FSM corresponds to the "brain" or algorithm controlling the agent.It contains a *state table* (also called *next state/output table*). Outputs are the actions and the next control state. Inputs are the control state s and defined input situations x. 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 the agents' behavior. 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 pro- and reactive activity). Here the intelligence is limited and realized by a mapping of its state and inputs to the next state, actions and outputs. Actions and outputs that an agent is able to perform are:

- **nextstate**: $state \leftarrow nextstate \in \{0, \ldots, N_{states} - 1\}$.
- **move**: $move \in \{0, 1\} \equiv \{wait, go\}$.
- **turn**: $turn \in \{0, 1, 2, 3\}$.
  The new direction is $D(t+1) \leftarrow (D(t) + turn)\ mod\ 4$.
- **setcolor**: $setcolor \in \{0, 1, 2, 3\}$.
  The new color is $L(t+1) \leftarrow (L(t) + setcolor)\ mod\ 4$.
- **setmarker**: The new marker is $M(t+1) \leftarrow setmarker \in \{0, 1\}$.
- **signal**: An agent emits a $signal \in \{0, 1\}$ that can be used by another agent.

An agent shall react on the following inputs:

- its **own control state**,
- its **own direction**,
- the **cell's color and marker** it is situated on,
- a **border cell** in front,
- a **blocked** situation/condition, caused either by a border, another agent in front, or when another prior agent can move to the front cell in case of a conflict. The inverse condition is called *free*.
- a **signal** from another agent if it stays in front.

An agent has a moving direction D that also selects the cell in front as the actual neighbor. What can an agent directly observe from a neighboring agent? In the used model it can only register the presence of an agent in front and the emitted signal from that agent. So only a small part of the state of that agent is revealed in the form of a *signal*, which can be used for cooperation.

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 the agent cannot move and has to wait, but still it can turn and change the cell's color and marker. 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, $x \in \{0, 1, \ldots, 12\}$:

$x = 0$, if the agent is *blocked* by a border cell in front,
$x = 1 + (L - D) mod\ 4 + 4(marker)$, if the agent is *not blocked*,
$x = 9 + 2(signal)$, if $L \in \{0, 2\}$ and the agent is *blocked* by an agent in front,
$x = 10 + 2(signal)$, if $L \in \{1, 3\}$ and the agent is *blocked* by an agent.

This mapping was designed by experience and experiments. Of course, other input mappings could be defined, with more or less x codes, or other assignments, e.g. the color or marker in front of the agent or in its neighborhood could be taken into account. Note that agent's view is very limited, it can react only the cell data (control state, direction, color, marker) where it is situated on, and sometimes on the signal from an agent in front. Therefore the agent's task is really difficult to solve. (Imagine you are the agent, moving around in a dark space where you can only observe the color and marker on the ground, and sometimes you detect a border or a signal in front!): the larger the agent's view, the easier the task can be solved.

The used updating scheme is *synchronous*; An important issue is that asynchronous updating is more natural, as studied in [29,30]; see further Sect. 5.5.

4 Evolving FSMs by a Genetic Algorithm

An ultimate aim could be to find an FSM that is optimal for all possible initial configurations on average. This aim is very difficult to reach because it needs an 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 *specialist* optimized for (i) a fixed field size of $N = n \times n$, $n = 8$, (ii) a fixed number of agents k and (iii) 100 initial random configurations (for training and evaluation).

The number of different FSMs 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 states to $|s| = N_{states} = 8$, and the number of inputs to $|x| = 13$. A relatively simple genetic algorithm similar to the one in [7] was used in order to find (sub)optimal FSMs with reasonable computational cost. A possible 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), setcolor(j), setmarker(j), signal(j))*.

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 FSM_i on a given initial configuration. Then the mean fitness $\overline{F}(FSM_i)$ is computed by averaging over all initial configurations of the training set. $\overline{F}$ is then used to rank and sort the FSMs.

Evolved Finite State Machines. 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 agent's task in relation to their capabilities. In addition the search space is very large and difficult to explore. The total computation time on a Intel Xeon QuadCore 2 GHz was around 3 weeks to find the needed FSMs.

Let the best found FSM for k agents and a reached order of h be denoted by FSM(k,h). At first the FSM(k=16, h=18) was evolved and used in Sects. 5.1 and 5.2; then FSM(1,18) in Sect. 5.3 and finally FSM(16,12), FSM(32,12), FSM(48,12) in Sect. 5.4.

5 Simulations and Evaluations

5.1 Simulation

The whole multi-agent system using the evolved FSM(16, 18) was simulated (Fig. 3). The snapshots (a) show how the line patterns are being built. The communication by signals is shown in Fig. 3b. It can be seen that at the end long lines in four colors appear. The markers (depicted as green or red small squares) are random at the beginning and almost all of them are changed during the run by the agents. The information stored in the markers is used for feedback and indirect communication. Although it was not detectable in which way, experiments without markers showed that the performance is much lower, a result also found in [8,11]. Note that a formed line pattern is not stable. After having

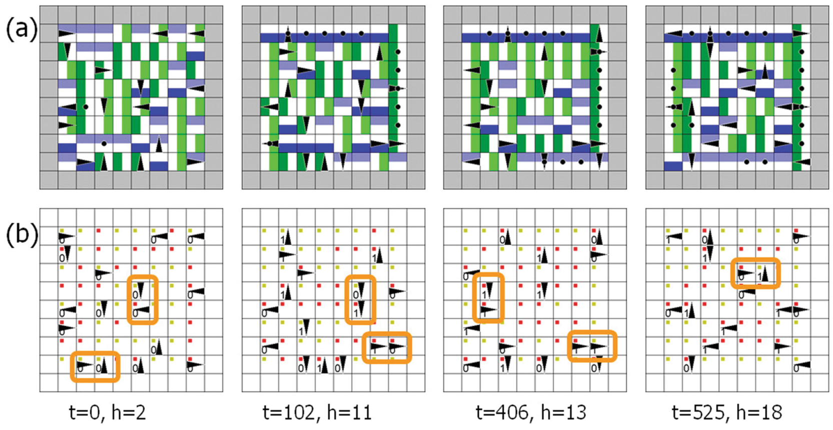

Fig. 3. (a) Snapshots showing how a line pattern is formed in a 8×8 field by 16 agents starting from a random configuration. h = degree of order.Agents are represented by black triangles, spins by colors, and markers by small squares (red = 1, yellow = 0). Dots represent template hits. (b) The signals that the agents emit are shown, colors and hits are hidden. When an agent is blocked by another agent in front, it can read its signal. Some communication situations by signals are encircled in orange. (Color figure online)

reached the required degree of order the pattern is changing and the degree of order is fluctuating. Detecting a certain degree of order in a decentral way for termination is a seperate issue. It could be achieved by all-to-all communication of the hits.

5.2 Effectiveness Test

At first FSM($k = 16$, $h = 18$) was investigated. The chosen density of the agents was $\delta = 1/4$, because from [7] this density turned out to be most cost-effective in terms of $time \times agents$. The degree of order to be reached was incremented from lower levels until $h_{target} = 18$, the theoretical maximum is at least $h_{max} = (n-2)^2 = 36$. The aimed relative degree of order was $h_{rel} = 50\,\%$, which is relatively low. The problem was that it was very difficult to reach a higher degree for this task with the given features of the agents. The found FSM was successful on all given 100 initial configurations of the test and evaluation set. The mean and extrema times to order a field were $t_{mean} = 1079 \pm 1011$, $t_{min} = 73$, $t_{max} = 5365$.

Then this FSM was simulated for another number of agents and it was expected that it is also 100 % successful, but that was not the case. Therefore FSM(16, 18) was tested with a lower order to be reached, $h = 12$ instead of $h = 18$. The result of this test is shown in Fig. 4a, success rate *vs.* number of agents. The success rate is the number of fields out of 100 which were ordered with degree 12. This diagram shows that this FSM is working effectively (satisfactorily successful) for a number of agents nearby 16, the number it was evolved for. This is an argument that a specialist was evolved.

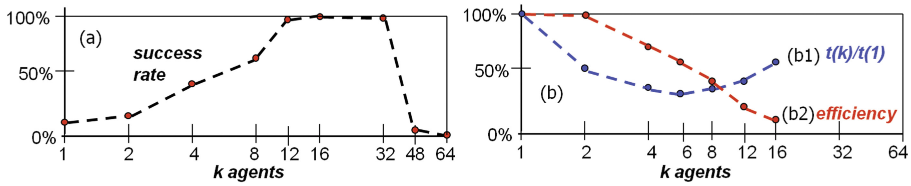

Fig. 4. (a) Effectiveness test. The top evolved FSM($k = 16$, $h = 18$) for size $= 8 \times 8$ was used. Then it was tested on how many fields it was successful for a different number k of agents, with a lower $h = 12$. This FSM is only effective for a number of agents nearby 16. (b) The top evolved FSM($k = 1$, $h = 18$) was used then tested with $h = 12$. Successful for 100 fields in the range $k = 1, \ldots, 16$. (b1) Mean time ratio, minimal for $k = 6$. (b2) The efficiency decreases continuously with the number of agents.

5.3 FSM Evolved for One Agent

Mean and extrema times for FSM(1, 18) to order a field were $t_{mean} = 2272 \pm 1666$, $t_{min} = 260$, $t_{max} = 7828$. Now the question is, how does FSM(1, 18), evolved for one agent, behaves if the number of agents is varied. Time-scaling and efficiency were evaluated for a lowered degree of order $h = 12$; time $t(k)$ was normalized to $t(1)$. The graph (b1) in Fig. 4b shows that the minimal time is reached for $k = 6$, whence a density of agents $\delta = 6/64 = 9.4\,\%$. This result was not expected, because in [7] the optimal density was 25 %. An explanation could be that this task is more difficult, and therefore each agent needs a larger "personal" local memory (more markers and colors) to store intermediate ordering information. The graph (b2) shows the *efficiency*, i.e. the speedup divided by the number of agents: $(t(1)/t(k))/k$. The decreasing efficiency with the number of processors is quite common in multiprocessor systems, as we can interpret agents as moving processors.

5.4 Effect of Signals

The question is how much the communication with signals speed up the task. A scientifically sound answer goes beyond the scope of this paper. Nevertheless a partial answer can be given for some test cases. For that reason six top FSMs were evolved with order $h = 12$, for $k = 16, 32, 48$, and with (resp. without) signal. The following values were obtained:

no. of agents k	16	32	48
t1 = t(with signal)	111	185	1553
t2 = t(without signal)	134	237	2146
t1/t2	.83	.78	.72

This evaluation shows that the time ratio t_1/t_2 is significantly lower than 1 (signals are efficient) and it increases with the number of agents when more communication is probable. Another result is that times t_1 and t_2 increase with k in this interval; this confirms the observation from the previous cases, that a high density of agents is not efficient.

5.5 Updating-Tolerance

Further simulations were done in order to show the effect of asynchronous updating. The optimal FSM(16, 18) evolved under synchronous updating was used. Then each generation was divided into $k = 16$ subgenerations. For each subgeneration one active agent was selected at random. The time counter was incremented by $1/k$ for each subgeneration, i.e. by one for one generation. 10 runs with different random seeds were performed on the given 100 initial fields. The mean and extrema times to order a field are $t_{mean} = 859.13 \pm 856.78$, $t_{min} = 34.25$, $t_{max} = 5628.94$. Compared to synchronous updating (Sect. 5.2), the t_{mean} ratio $t_{asyn}/t_{syn} = 859.13/1079 = 0.8$. This means that asynchronous updating was successful on all fields using the same synchronous rule, and the time was even 20 % lower for this test case.

6 Conclusion

The objective was to find FSM-controlled agents that can form specific line patterns. The class of path patterns was defined by a set of templates, small

3×3 local patterns. For 8×8 fields, several FSM were evolved with a different number of agents and a certain degree of order h. The agents are able to form successfully the aimed line patterns with the predefined degree of order by means of markers and signals. The FSM evolved for 16 agents is only effective for an agent number nearby 16. The FSM evolved for 1 agent is effective for up to 16 agents and most efficient for 6 agents. Signals speed-up the task significantly, especially for a higher density of agents when the communication probability is higher. The general result is that the generation of specific patterns by CA agents can be designed in a methodical way.

References

1. Deutsch, A., Dormann, S.: Cellular Automaton Modeling of Biological Pattern Formation. Birkäuser, Basel (2005)
2. 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)
3. Itoh, M., Takahira, M., Yatagai, T.: Spatial arrangement of small particles by imaging laser trapping system. Opt. Rev. **5**(1), 55–58 (1998)
4. Jiang, Y., Narushima, T., Okamoto, H.: Nonlinear optical effects in trapping nanoparticles with femtosecond pulses. Nat. Phys. **6**, 1005–1009 (2010)
5. Roberts, Jr., G.: X-ray laser explores how to write data with light. National Accelerator Laboratory, 19 March 2013. https://www6.slac.stanford.edu/news
6. 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)
7. Hoffmann, R.: How agents can form a specific pattern. In: Wąs, J., Sirakoulis, G., Bandini, S. (eds.) ACRI 2014. LNCS, vol. 8751, pp. 660–669. Springer, Heidelberg (2014)
8. Hoffmann, R.: Cellular automata agents form path patterns effectively. Acta Phys. Pol. B Proc. Suppl. **9**(1), 63–75 (2016)
9. Halbach, M., Hoffmann, R., Both, L.: Optimal 6-state algorithms for the behavior of several moving creatures. In: El Yacoubi, S., Chopard, B., Bandini, S. (eds.) ACRI 2006. LNCS, vol. 4173, pp. 571–581. Springer, Heidelberg (2006)
10. Ediger, P., Hoffmann, R.: Optimizing the creature's rule for all-to-all communication. In: Adamatzky, A., Alonso-Sanz, R., Lawniczak, A., (eds.) Automata-2008: Theory and Applications of Cellular Automata, pp. 398–412 (2008)
11. 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)
12. Hoffmann, R., Désérable, D.: All-to-all communication with cellular automata agents in 2D grids. J. Supercomp. **69**(1), 70–80 (2014)
13. Ediger, P., Hoffmann, R.: CA models for target searching agents. Elec. Notes Theor. Comp. Sci. **252**, 41–54 (2009)
14. Ediger, P., Hoffmann, R., Désérable, D.: Routing in the triangular grid with evolved agents. J. Cell. Automata **7**(1), 47–65 (2012)
15. Ediger, P., Hoffmann, R., Désérable, D.: Rectangular vs triangular routing with evolved agents. J. Cell. Automata **8**(1–2), 73–89 (2013)

16. 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)
17. Mesot, B., Sanchez, E., Peña, C.-A., Perez-Uribe, A.: Artificial Life VIII. SOS++: Finding Smart Behaviors Using Learning and Evolution. MIT Press, Cambridge (2002)
18. Blum, M., Sakoda, W.J.: On the capability of finite automata in 2 and 3 dimensional space. In: SFCS 1977, pp. 147–161 (1977)
19. Bonabeau, E.: From classical models of morphogenesis to agent-based models of pattern formation. Artif. Life **3**(3), 191–211 (1997)
20. Hamann, H.: Pattern Formation as a Transient Phenomenon in the Nonlinear Dynamics of a Multi-agent System. MATHMOD, Vienna (2009)
21. Nagpal, R.: Programmable pattern-formation and scale-independence. In: Minai, A.A., Bar-Yam, Y. (eds.) Unifying Themes in Complex Systems IV: Proceedings of the Fourth International Conference on Complex Systems, pp. 275–282. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). http://dx.doi.org/10.1007/978-3-540-73849-7_31
22. Yamins, D., Nagpal, R.: Automated global-to-local programming in 1-D spatial multi-agent systems. In: Proceedings of 7th International Conference AAMAS, pp. 615–622 (2008)
23. Bandini, S., Vanneschi, L., Wuensche, A., Shehata, A.B.: A neuro-genetic framework for pattern recognition in complex systems. Fund. Inf. **87**(2), 207–226 (2008)
24. Hoffmann, R.: The GCA-w massively parallel model. In: Malyshkin, V. (ed.) PaCT 2009. LNCS, vol. 5698, pp. 194–206. Springer, Heidelberg (2009)
25. Hoffmann, R.: Rotor-routing algorithms described by CA-w. Acta Phys. Pol. B Proc. Suppl. **5**(1), 53–67 (2012)
26. 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, Switzerland (2015)
27. Hardy, J., Pomeau, Y., de Pazzis, O.: Time evolution of a two-dimensional classical lattice system. Phys. Rev. Lett. **31**(5), 276–279 (1973)
28. Achasova, S., Bandman, O., Markova, V., Piskunov, S.: Parallel Substitution Algorithm - Theory and Application. World Scientific, Singapore (1994)
29. Bouré, O., Fatès, N., Chevrier, V.: Probing robustness of cellular automata through variations of asynchronous updating. Nat. Comp. **11**(4), 553–564 (2012)
30. Bandini, S., Bonomi, A., Vizzari, G.: An analysis of different types and effects of asynchronicity in cellular automata update schemes. Nat. Comput. **11**(2), 277–287 (2012)