CA Agents for All-to-All Communication Are Faster in the Triangulate Grid

Rolf Hoffmann[1] and Dominique Désérable[2]

[1] Technische Universität Darmstadt, FB Informatik, FG Rechnerarchitektur, Hochschulstraße 10, 64289 Darmstadt, Germany
hoffmann@ra.informatik.tu-darmstadt.de
[2] Laboratoire LGCGM UPRES EA 3913, Institut National des Sciences Appliquées, 20 Avenue des Buttes de Coësmes, 35043 Rennes, France
deserabl@insa-rennes.fr

Abstract. The objective was to find the behaviour of agents to solve the all-to-all-communication task in the cyclic triangulate and square grids in shortest time. The agents should be reliable, meaning that they are successful on almost any initial configuration. In order to solve the problem, the multi-agent system was modeled by Cellular Automata with synchronous updating, and the behavior of the agents was modeled by an embedded finite state machine (FSM). Agents can move or stay, and turn to any direction. An agent is able to leave a trace by setting a color flag on its site. Colors allow indirect communication similar to pheromones, speed up the task and contribute to a better reliability. More reliable agents could be found by using different initial control states for the agent's FSMs. A simple genetic procedure based on mutation was used to evolve near optimal FSMs for the triangulate and square grid. Agents in the triangulate grid can solve the task in around 2/3 of the time compared to agents in the square grid. The communication time depends also on the density of agents in the field, e.g. agents with density 4/(16 x 16) turned out to be the slowest.

Keywords: All-to-All communication, Cellular Automata, Multi-agent-system, Evolving Finite State Machines, Square and Triangulate Grids, Indirect Communication.

1 Introduction

Problem and Objectives. Several agents that are moving around in a Cellular Automata grid are able to solve the all-to-all communication task. Initially each one has got a mutually exclusive part of the whole information which can be distributed when the agents meet locally. The task is considered *successful* when all agents have gathered the complete information. Information parts already gathered by an agent can be handed over to another agent. The objectives here are:

- find reliable agents for the square grid (S-grid "S") and the triangulate grid (T-grid "T")
- compare the communication time in S and T for a varying density of agents.

V. Malyshkin (Ed.): PaCT 2013, LNCS 7979, pp. 316–329, 2013.

The general goal of our project is to find systematically the local behavior of moving agents in a multi-agent system modeled by Cellular Automata in order to fulfil a given global task.

Previous and Related Work. All-to-all communication (A2A) is a very common task in distributed computing. A2A in multi-agent systems is related to multi-agent problems like finding a consensus [1], synchronizing oscillators, flocking theory or rendezvous in space [2], or in general to distributed algorithms with robots [3]. The problem's specification can depend on many fixed or dynamic varying parameters like the number and location of nodes, the number and location of processes, the number, users and properties of the communication channels and so on.

In former investigations [4] we have tried to find the best algorithms for the *Creatures' Exploration Problem*, in which the creatures (agents) have the task to visit all empty cells in shortest time. The problem herein is related to it with respect to finding an optimal movement of the agents. But this task is different: now the agents shall exchange all their individual information in shortest time.

We have already studied A2A [5–9] in the S-grid. In these investigations, environments were used with and without border (cyclic), with and without obstacles, with and without colors, and with different local communication situations. The main results were

- environments with border are easier (faster) to solve
- colors speed up the task by a factor of around 2
- time-shuffling (alternating two FSMs in time) speeds up the task
- for a field of size 33 x 33 and 16 agents, the best reached communication time was
  - 406 time steps, using actions turn right/left and move, time-shuffling two FSMs with 6 states each, no coloring [8]
  - 195 time steps, using actions turn right/left/straigth/back and move, coloring, color-FSM and action-FSM with 8 states each [9]
  - 320 time steps, using actions turn right/left/straigth and move, coloring, 6-state FSM [7].
- the evolved agents (FSMs) were not always reliable.

The former modeling and parameter setting differs in many details from the one used here. Based on this experience we defined a new, more clear modeling (especially with a von-Neumann communication) to be used for S and T grids.

Our research in general is also related to works like: evolving optimal rules for cellular automata (CA) [10, 11], finding out the center of gravity by marching pixels by evolutionary methods [12], modeling multi-agent systems in CA to simulate pedestrian movement [13], or traffic flow [14].

Contribution and Organization. The main contribution is the finding of reliable, near optimal agents controlled by a finite state machine solving the A2A task in the square and triangulate grids, and showing that agents in the triangulate grid are around 1.5 times faster than in the square grid.

Topology and basic communication properties of the square and triangular grids are given in Sect. 2. The modeling of the multi-agent system (MAS) is presented in Sect. 3. The genetic procedure used to evolve the agents' behavior is described in Sect. 4. Sect. 5 provides the results of this investigation and Sect. 6 concludes.

2 Topology of the CA Networks: S-Grid and T-Grid

Consider the square blocks in Fig. 1 with $N = 2^n \times 2^n$ nodes where n will denote the "size" of the networks. The nodes are labeled according to the XY–orthogonal coordinate system. In the left block, a node (x, y) is connected with its four neighbors $(x \pm 1, y)$, $(x, y \pm 1)$ (with addition modulo 2^n) respectively in the N–S, W–E directions, giving a 4–valent torus usually denoted as "square" and labeled "S" or "S-grid" in the sequel. In the right block, two additional links $(x-1, y-1)$, $(x+1, y+1)$ are provided in the diagonal NW–SE direction, giving a 6–valent torus usually denoted as "triangulate" and labeled "T" or "T-grid" in the sequel [15]. Because their associated graphs are regular their number of links is, respectively, $2N$ for torus S and $3N$ for torus T. Both networks are scalable in the sense that one network of size n can be built from four blocks of size $n-1$. To be precise, let us finally note that the dual *cellular* tilings S^* and T^* as displayed in Fig. 2 and associated to S and T, are respectively the $\{4, 4\}$ square tiling and the $\{6, 3\}$ (homeomorphic) honeycomb, where $\{p, q\}$ is the Schläfli symbol.

The basic routing schemes are driven by the Manhattan distance in S and by the so-called "hexagonal" distance in T [16]. Global communications such as "One-to-All" broadcasting or "All-to-All" gossiping are frequently used in parallel applications: respectively, *one* node diffuses a message to *all* nodes (broadcasting) whereas *all* nodes diffuse their own message to *all* nodes (gossiping) [17]. For a given topology, their exists at least one deterministic protocol for each global communication. But in the context of multi-agent systems herein the context is quite different, because on one hand the number of agents is not

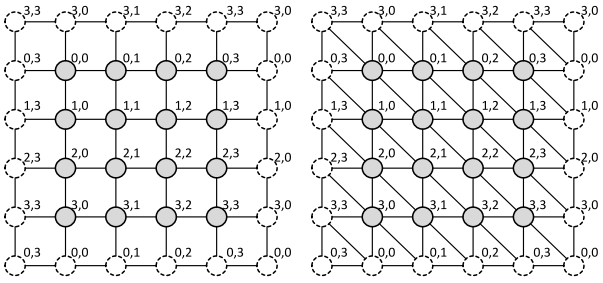

Fig. 1. Tori "S" and "T" of size $n = 2$, of order $N = 16$, labeled in the XY coordinate system; their number of links is $2N$ for S and $3N$ for T

necessarily the number of nodes and on the other hand agents' trajectory is not deterministic.

Two important parameters for the routing task in the networks are the diameter and the mean distance. The diameter defines the shortest distance between the most distant pair of nodes and provides a lower bound for routing or other global communications; such a pair is said to be antipodal. The mean distance gives an average for the performance of the routing. Diameter D_n and mean distance $\bar{\delta}_n$ are given by [18]

$$D_n^S = \sqrt{N}\ ;\ D_n^T = \frac{2(\sqrt{N}-1)+\varepsilon_n}{3} \tag{1}$$

$$\bar{\delta}_n^S = \frac{\sqrt{N}}{2}\ ;\ \bar{\delta}_n^T \approx \frac{1}{6}\left(\frac{7\sqrt{N}}{3} - \frac{1}{\sqrt{N}}\right) \tag{2}$$

where $\varepsilon_n = 1$ (resp. 0) depends on the odd (resp. even) parity of n and where the upper symbol identifies the torus type; whence the ratios denoted by

$$D_n^{T/S} \approx 0.666\ ;\ \bar{\delta}_n^{T/S} \approx 0.775 \tag{3}$$

between diameters and mean distances. Fig. 2 highlights the distances from an arbitrary cell, and thereby the diameters, in this family of CA networks of size 3.

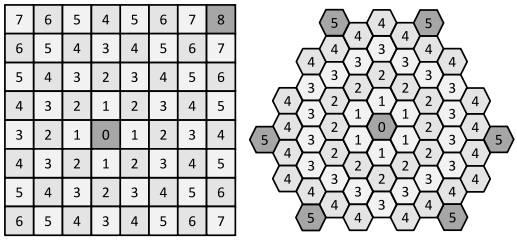

Fig. 2. Distances and antipodals from a center cell in the cellular representation of S and T for $n = 3$: $D_3^S = 8$, $\bar{\delta}_3^S = 4$; $D_3^T = 5$, $\bar{\delta}_3^T \approx 3.09$

3 CA Modeling of the Multi-agent System

The whole system is modeled by cellular automata (CA). It consists of an environment ($M \times M$ S- or T-grid, where $M = 2^n$ herein) without borders (cyclic wrap-around) and k uniform agents. We decided not to use borders because this case is more difficult to solve because the agents cannot use the borders as an orientation where they can meet. The state of CA cell is either *empty* or the state of an agent (if an agent is situated on that site).

Agent's State. The *state* of an agent is:

$$\texttt{state} = \{\texttt{IDentifier}, \texttt{Direction}, \texttt{ControlState}, \texttt{CommunicationVector}\}.$$

The agents are distinguished by their identifier $\texttt{ID} \in \{0 \ldots N_{agents} - 1\}$. Thereby the agent's trajectories can be traced, and the ID can be used for resolving conflicts and as an information to vary the initial control state. The moving direction is $\texttt{Direction} \in \{0..3\}$ for S and $\texttt{Direction} \in \{0..5\}$ for T, respectively. The `ControlState` is the state of an embedded finite state automaton, controlling the actions.

Communication Method. In order to model the distribution of information, a bit vector `CommunicationVector` of k length is stored in each agent. At the beginning the bits are set mutually exclusive ($\texttt{bit}(i)$=1 for agent(i)). Agents exchange their information when they meet in a certain communication situation. We decided that an agent can read all the information from the other agents from its nearest neighbours (4 in S and 6 in T). This communication situation does not depend on the agents moving direction, as in the communication situations we used earlier. We think that this new definition is more simple and will yield similar results. The exchange of information is modeled by simply OR-ing the communication vectors of the involved agents. The task is successfully solved when the k–vector becomes $(11 \ldots 1)$ for all agents.

Actions. An agent can perform the three basic actions `move`, `turn`, `setcolor` independently of each other.

- The agent moves in the current direction if `move`=1 when it can move (free, not blocked), otherwise it waits. The agent waits unconditionally if `move`=0.
- For the S-grid, the basic action $\texttt{turn} \in \{0, 1, 2, 3\}$ defines the new direction: $direction \leftarrow direction + \texttt{turn} \times 90°$.
  This means that the agent can turn to any of the four directions.
  For the T-grid, the basic action $\texttt{turn} \in \{0, 1, 3, 5\}$ defines the new direction: $direction \leftarrow direction + \texttt{turn} \times 60°$.
  This means that the T-agent cannot turn to $\pm 120°$. This decision was made because the cardinality of `turn` should be the same, in order to have the same complexity of abilities for the S- and T-agents.
- Apart from the agent's movement and the information exchange, an agent has indirect communication capabilities. Each cell of the environment contains a *color* (status flag) which is either 0 or 1 and used as an input for the decision making process. The color can be seen as a tracing information like a "pheromone" left by other agents or even by the reading agent itself. The agent is able to change the color of the cell on which the agent is currently located:
  $color \leftarrow 0$ if `setcolor`=0, and $color \leftarrow 1$ if `setcolor`=1.

Thus in total there are 16 possible actions that an agent can perform. The actions can be written in abbreviated form, using $\texttt{turn} \in \{S, R, B, L\}$ (Straight, Right, Back, Left), $\texttt{move} \in \{m, .\}$ (move, wait) and $\texttt{setcolor} \in \{0, 1\}$. Thus the action set is:

{*Sm0, Sm1, S.0, S.1, Rm0, Rm1, R.0, R.1*
Bm0, Bm1, B.0, B.1, Lm0, Lm1, L.0, L.1}.

Input Information. The information on which an agent can react on, is

- the color of the current cell the agent is situated on,
- the color of the cell ahead (also called the *front cell*),
- if there is an agent in front or not,
- if there are agents that want to move to the same front cell (conflict),
- the agent's own control state.

Conflicts. An agent cannot move if it detects an agent in front or is a looser of the conflict resolution protocol. A conflict occurs when two or more agents want to move to the same front cell (cell in conflict). In order to detect a conflict an extended neighborhood [5] is needed (Manhattan distance 2 in the moving direction). Alternatively, especially when a fast hardware implementation is pursued, the conflict detection can be performed by an arbitration logic [4] available in each cell. The arbitration logic evaluates the move requests coming from the agents and replies asynchronously by an acknowledge signal in the same clock cycle. In order to resolve a conflict, a resolution strategy has to be defined. We defined that the agent with the lowest ID has priority.

Control FSM. The decision upon which action will be performed depends on the behavior of the agent. The behavior (algorithm) of an agent is defined by a finite state machine (FSM) of type MEALY. A FSM contains a state register and a transition/output table. Input of the table is the current input x and the current state s, output is the next state s' and current output y, e.g. Fig. 3.

Input x of the concrete FSM used here is the own control state s of the FSM, the move condition $x = \texttt{canmove} \in \{0, 1\}$, the color of the own cell the agent is situated on, and the color of the front cell. The inverse move condition is called *blocked*. Thus altogether there are 8 different input values. The move condition is computed by a separate function, that evaluates to TRUE if (i) there is no agent in front and (ii) in case of conflict the own ID is the lowest compared to the others.

Output of the FSM is $y = (\texttt{move}, \texttt{turn}, \texttt{setcolor})$.

In order to keep the control automaton simple, we restrict the number of states and actions to a certain limit (see Sect. 4). To solve the problem very general either theoretical or practical with respect to all interesting parameters is too difficult. Therefore we have specialized our investigation. The grid size was set to 16×16. The number of agents was set to $k = 16$ for the genetic procedure but then varied from $k = 2$ to the maximum (number of cells).

4 The Genetic Procedure

The ultimate goal is to find the optimal behavior on average for all possible initial configurations in S and T. As we cannot test all possible configurations, we restrict our investigation to a field size of 16 x 16, with a certain number of agents

```
S-agent FSM /x = 0\  /x = 1\  /x = 2\  /x = 3\  /x = 4\  /x = 5\  /x = 6\  /x = 7\
blocked        0        1        0        1        0        1        0        1
color          0        0        1        1        0        0        1        1
frontcolor     0        0        0        0        1        1        1        1
state       0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3
|nextstate  2 3 1 1  0 3 3 2  1 3 0 2  0 0 2 1  1 2 2 0  2 3 2 0  2 2 3 0  3 1 0 2 |
|setcolor   1 1 0 0  0 1 0 1  0 0 0 1  1 0 1 1  0 0 0 0  0 0 0 1  0 0 0 1  1 0 0 0 |GENOM
|move       1 1 0 1  0 1 1 1  1 1 1 1  1 1 1 0  1 1 1 1  0 0 0 0  0 0 0 1  0 1 0 0 |S-agent
|turn       3 0 1 0  1 1 1 2  3 0 0 3  2 1 2 3  0 1 2 1  3 0 1 3  2 3 3 3  3 2 2 3 |
index i     0 1 2 3  4 5 6 7  8 91011 12131415 16171819 20212223 24252627 28293031
```

Fig. 3. A state table for an FSM controlled S-agent moving and communicating in the S-grid. The `turn` actions for the S-agent mean: *turn* $0°/90°/180°/-90°$ for (`turn` = 0,1,2,3) – This FSM represents also the best found algorithm for the S-agents

$N_{agents} \in \{2, 4, 8, 16, 32, 64, 256\}$. We will be satisfied if we can find near optimal (fast) agents, that are also reliable, e.g. which are able to solve the problem for each N_{agents} and for all initial configurations out of a set of $N_{fields} = 1003$. Thus 1000 initial configurations were randomly generated (position, direction) for each N_{agents}, plus 3, manually designed. The manually designed are difficult to be solved by simple uniform agents, because it may easily happen that agents never meet (when agents follow synchronously the same strategy). The first is a queue of N_{agents}, all agents with direction $\rightarrow$; the second is a queue of N_{agents}, all agents with direction $\leftarrow$; and in the third configuration the agents are placed in the diagonal with maximum space between them, all agents with direction $\leftarrow$.

As the search space for different behaviors is very large we are not able to check all possible behaviors by enumeration. The number of state machines which can be coded using a state table is $K = (|s||y|)^{(|s||x|)}$ where $|s|$ is the number of states, $|x|$ is the number of different input values and $|y|$ is the number of different outputs. As the search space increases exponentially we used a genetic procedure in order to find the best behavior within a reasonable computational time limit. Nevertheless the number of states and inputs has to be kept low in order to find a good solution in reasonable time.

The concatenation of the (`nextstate`, action)-pairs (s', y) for all input combinations with index i (`state` table in Fig. 3) defines the genome of one individual, a possible solution.

One population of N individuals is updated in each generation (optimization iteration). During each iteration $N/2$ offsprings are produced from the top $N/2$ individuals. The union of the current N individuals and the $N/2$ offsprings are sorted according to their fitness, duplicates are deleted and the number of individuals is reduced to the limit of N in the pool. In order not to get stuck in a local minimum and to allow a certain diversity in the gene pool, the first b individuals from the second half of the gene pool are exchanged with the last b individuals from the first half of the gene pool. We used $N = 20$ and $b = 3$, therefore the individuals 7, 8, 9 are exchanged with the individuals 10, 11, 12, where the individuals are numbered from 0 to $N - 1$.

We experimented with the classical crossover/mutation method. Then we found that mutation only gave us similar good results. So we used here only

mutation. It is subject to further research which heuristic is best to evolve state machines. In previous work we used also crossover and parallel populations, but at the moment we have no far-reaching comparisons between different heuristics to evolve state machines.

An offspring is produced by modifying separately the `nextstate` action, the `setcolor` action, the `move` action, and the `turn` action for each input combination (`index` in the FSM table):

`nextstate` $\leftarrow$ `nextstate`$+1 \mod N_{states}$ with prob. p_1, otherwise unchanged,
`setcolor` $\leftarrow$ `setcolor`$+1 \mod N_{setcolor}$ with prob. p_2, otherwise unchanged,
`move` $\leftarrow$ `move` $+ 1 \mod N_{move}$ with prob. p_3, otherwise unchanged,
`turn` $\leftarrow$ `turn` $+ 1 \mod N_{turn}$ with prob. p_4, otherwise unchanged.

We tested different probabilities, and we achieved good results with $p_1 = p_2 = p_3 = p_4 = 18\%$.

The fitness of a multi-agent system is defined as the number of steps which are necessary to distribute (all-to-all) the information, averaged over all initial configurations (positions and directions of the agents) in a certain set. As we are looking for reliable agents, the cardinality of the set has to be reasonably high in order to be relatively sure that the agents are successful for any given initial configuration. As the behavior of the whole system depends on the behavior of the agents, we search for the agents' state algorithms (FSMs) that can solve the problem with a minimum number of steps for a large number of initial configurations.

The fitness function F is evaluated by simulating the agent system. It reflects two aspects:

1. The number of agents N_{agents} which have gathered the complete information. If an agent has gathered the complete information it is *informed*. If all agents are informed, we characterize the agent system, respectively the algorithm, as *successful*. If the agents are successful on all given initial configurations then we characterize the agent system, respectively the algorithm, as *completely successful*.
2. The number of steps in the CA simulation to reach successfulness. We will call this value *communication time* t_{comm}.

The used fitness function integrates these aspects by choosing a weight W such that a dominance relation is formed:

$$F_i = W(N_{agents} - a_i) + t_{i,comm} \qquad W = 10^4$$

where a_i is the number of informed agents, and $t_{i,comm}$ is the communication time for an initial configuration i. The first term $(N_{agents} - a)$ reflects the number of agents that are not informed. It diminishes for a successful FSM and then the relation $F_i = t_{i,comm}$ holds. Note that a lower fitness value is better. The fitness F_i is computed for each simulated initial configuration i and then averaged over all initial configurations N_{fields} in the given set as

$$F = \sum F_i / N_{fields} \ .$$

The communication time depends on the size of the field, the number of agents, the algorithm (FSM), and the given field (initial configuration). For the investigated field size of 16 x 16 the expected communication time is lower than 100. During the genetic procedure a reasonable simulation time limit was used, e.g. $t_{max} = 200$.

```
T-agent FSM /x = 0\  /x = 1\  /x = 2\  /x = 3\  /x = 4\  /x = 5\  /x = 6\  /x = 7\
blocked        0        1        0        1        0        1        0        1
color          0        0        1        1        0        0        1        1
frontcolor     0        0        0        0        1        1        1        1
state       0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3  0 1 2 3
|nextstate  1 2 1 2  1 0 3 0  2 1 0 3  1 2 1 3  1 2 0 2  0 1 3 0  2 2 1 1  2 2 1 1 |
|setcolor   1 1 1 1  0 1 1 1  0 0 1 1  0 1 0 0  0 0 0 0  1 1 1 1  0 0 1 0  1 1 1 0 |GENOM
|move       1 1 1 0  1 0 0 0  1 1 1 1  0 1 1 1  1 1 1 0  1 0 0 0  1 1 1 0  1 0 1 1 |T-agent
|turn       0 0 1 0  3 2 2 2  3 0 0 1  0 0 3 3  1 0 1 2  3 3 0 1  3 0 1 3  2 0 2 3 |
```

Fig. 4. Best evolved FSM for the T-agent. The `turn` actions for the T-agent mean: *turn* $0°/60°/180°/-60°$ for (`turn` = 0,1 2,3).

The genetic procedure starts with $N = 20$ random FSMs. Usually there is no FSM in the initial population that is successful. After some generations, some successful FSMs are found. Then, after further generations, FSMs are expected to be evolved that are completely successful.

The genetic procedure was applied in the following way. Four independent optimization runs on 1003 initial configurations were performed, with field size 16 x 16 and $N_{agents} = 8$. Then the top 3 completely successful FSMs of each run (altogether 12) were also tested for $N_{agents} = 2, 4, 8, 16, 32, 256$, each on 1000 random initial configurations plus 3 extra manually designed (agents queueing in a line, agents on the diagonal). FSMs which were completely successful on all these configurations (1003 + 5 x 1003) were extracted and ranked. Then the best FSM was selected.

Former investigation showed that it is very difficult or impossible to find reliable agents which are successful on any given initial configuration, because agents can follow similar routes which are "parallel" and therefore never intersect. In general, a certain inhomogeneity (in space or in time) or even a randomness has to be introduced to break the symmetry. Some of the approaches to make the agents more reliable, are:

1. use coloring,
2. use random-like pattern of initial colors,
3. use different species (FSMs) of agents,
4. start the agents in different control states,
5. add obstacles.

We used the 4^{th} option. Experiments showed that we could not find uniform reliable agents when all FSMs started in control state 0 or 3. Recall that we use

only 4 control states. But we were able to find reliable agents, when we started some of the agents in state 0 and the others in state 1. We decided then to use the initial state = 0/1 for agents with even/odd ID.

Table 1. Communication time for N_{agents} in the T-grid and S-grid in a 16 x 16 field, averaged over 1003 initial configurations. The best found T-algorithm and best found S-algorithm were used.

N_{agents}	2	4	8	16	32	256
T-grid	58.43	78.30	58.68	41.25	28.06	9.00
S-grid	82.78	116.12	90.93	63.39	42.93	15.00
T/S	0.706	0.674	0.645	0.651	0.690	0.600

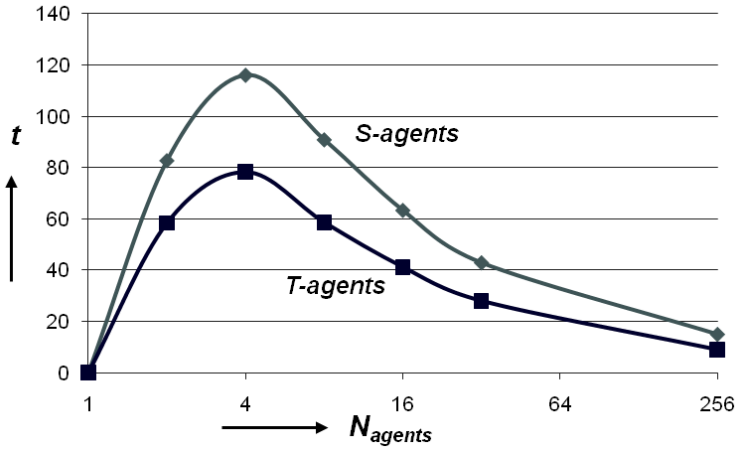

Fig. 5. Communication time for N_{agents} in the T-grid and S-grid in a 16 x 16 field. The T-agents are significantly faster than the S-agents, around 33 %. For $N_{agents} = 4$ maxima appear.

5 Comparison of the Evolved S- and T-Agents

The best found reliable FSM for the S-agent is shown in Fig. 3, and the best found T-agent is shown in Fig. 4. The communication time was evaluated by simulation for all $N_{agents} = 2, 4, 8, 16, 32, 256$, and for each case 1003 initial configurations. The agents were completely successful on all 5015 initial configurations using the same algorithm, one for the S-grid, one for the T-grid. The agents start in

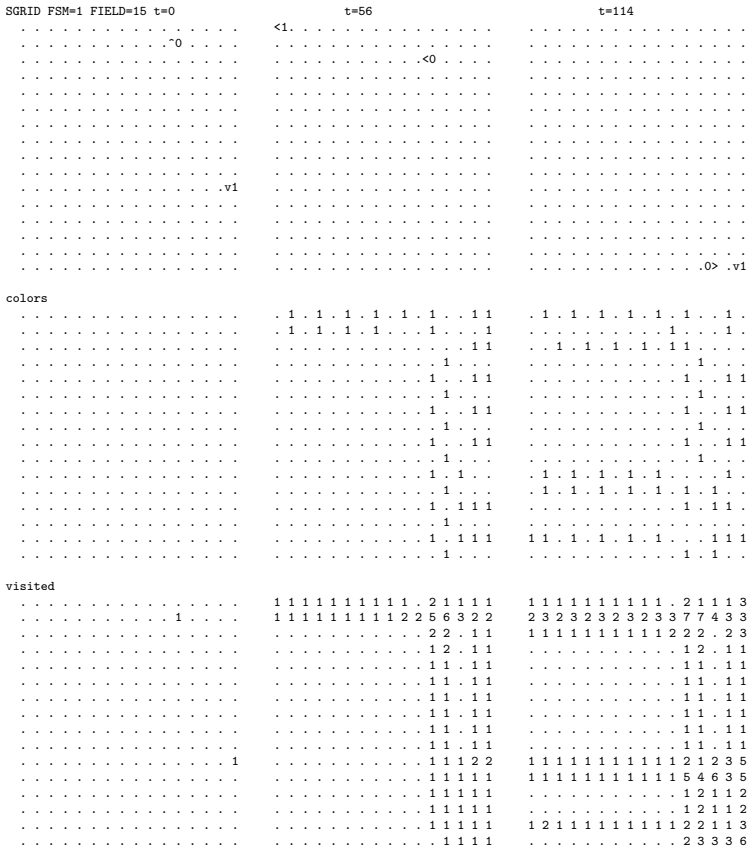

Fig. 6. Simulation of two agents in a 16 x 16 S-grid for a special initial configuration. The behaviour is defined by the best found FSM. Colors are set and reset (depicted in the middle): this information helps the agents to find each other faster. The agents build streets on which they travel more frequently (visited cells, depicted on the bottom). For this initial configuration, the S-agents need 114 time steps.

the initial control state `ID` mod 2; thereby the agents are very reliable. But we could not prove that these state machines will be successful for any arbitrary initial configuration.

Table 1 and Fig. 5 show the average communication time. It is interesting to observe that maxima were found for $N_{agents} = 4$, e.g. 4 agents communicate slower than 2 and 8 agents. Comparing T/S-agents, the ratio of communication time lies between 0.71 and 0.6, meaning that the T-agents are significantly faster. We expected a ratio of around 0.666, according to the diameter ratio $D^{T/S}$ in (3). All cases come close to this expected ratio. Note that the communication times in T and S are not related to the mean distance ratio $\bar{\delta}_n^{T/S}$.

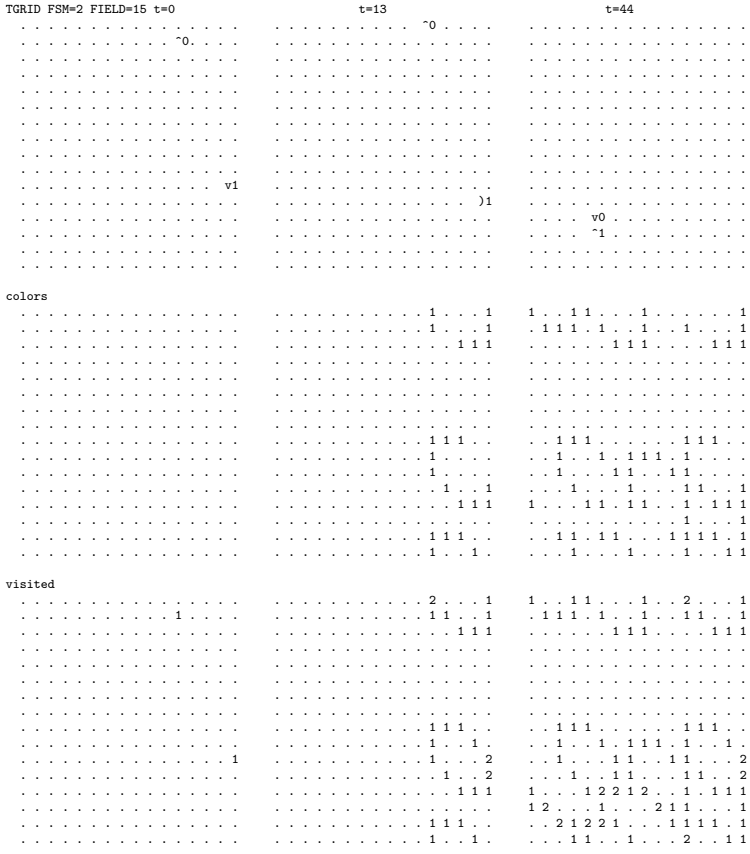

Fig. 7. Simulation of two agents in a 16 x 16 T-grid for a special initial configuration. The behaviour is defined by the best found FSM. Colors are set and reset (depicted in the middle): this information helps the agents to find each other faster. The agents build honeycomb-like networks on which they travel more frequently (visited cells, depicted on the bottom). For this initial configuration, the T-agents need only 44 time steps compared to 114 time steps for the S-agents.

In the special case $N_{agents} = 256$, the system is fully packed with agents that cannot move, only communicate. Then the communication time is the diameter (1) (the communication after the initial placement is not counted).

Two sample simulations show how the agents move and set colors in order to communicate. In the S-grid of Fig. 6, the agents build orthogonal communication streets (where the agents prefer to move) by the help of the colors. We can observe a few streets at time 56 (horizontal or vertical) and more at the end (t = 114).

For the T-agents in Fig. 7, we observe that they can build honeycomb-like networks! At time 13 we observe two honeycombs and at the end (t = 44) there are several of them. It has to be mentioned that the T-agents are faster on average, but for some initial configurations they can be slower.

In a previous work [9], 195 time steps were reached for a 33 x 33 S-grid with 16 agents. For comparison, our best 12 agents, evolved for a 16 x 16 grid with 8 agents, were tested on 1003 randomly generated fields of size 33 x 33 with 16 agents. The best S-agent needed 229 time steps and the best T-agent needed 181 time steps, and the agents were reliable. Again the T-agent is faster than the S-agent. However, our T-agents are not so fast as the ones evolved in [9]. The reasons are: (1) we used only one FSM with 4 states, instead of using two FSMs with 8 states each, (2) we did not specifically evolve our agents for the field size of 33 x 33, and (3) our agents were specifically evolved for a high reliability (agents start in different initial control states).

6 Conclusion

The multi-agent system needed for simulation and optimization was described by Cellular Automata, and the agent's behavior was modeled by a finite state machine (FSM). For the triangulate and square grid, near optimal FSMs were evolved by a genetic procedure. In order to make the agents more reliable (successful on any initial configuration), half of the agents start in state 0, the other half in state 1. Thereby the agents could solve the task successfully for a large number (5015, each for the T- and S-grid) of initial configurations. T-agents can solve the task in about 2/3 of the time the S-agents needed. For the 16 x 16 grid, two and more than 4 agents can communicate faster than 4 agents. In further work it could be studied how fast and reliable agents are when using more states, more colors, obstacles, or borders.

References

1. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE 95(1), 215–233 (2007)
2. Lin, J., Morse, A.S., Anderson, B.D.O.: The Multi-Agent Rendezvous Problem. An Extended Summary. In: Kumar, V., Leonard, N., Stephen Morse, A. (eds.) Cooperative Control. LNCIS, vol. 309, pp. 257–289. Springer, Heidelberg (2005)
3. Santoro, N.: Distributed Algorithms for Autonomous Mobile Robots. In: Navarro, G., Bertossi, L., Kohayakawa, Y. (eds.) TCS 2006. IFIP, vol. 209, p. 11. Springer, Boston (2006)
4. 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)
5. Ediger, P., Hoffmann, R.: Optimizing the creature's rule for all-to-all communication. In: EPSRC Workshop Automata-2008, Theory and Applications of Cellular Automata, Bristol, UK, pp. 398–412 (2008)
6. Hoffmann, R., Ediger, P.: Evolving multi-creature systems for all-to-all communication. In: Umeo, H., Morishita, S., Nishinari, K., Komatsuzaki, T., Bandini, S. (eds.) ACRI 2008. LNCS, vol. 5191, pp. 550–554. Springer, Heidelberg (2008)
7. 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)

8. Ediger, P., Hoffmann, R.: Evolving hybrid time-shuffled behavior of agents. In: 13th Workshop on Nature Inspired Distributed Computing NIDISC, Proc. Par. Dist. Proc. IPDPS, pp. 1–8 (2010)
9. Ediger, P., Hoffmann, R.: All-to-all communication with CA agents by active coloring and acknowledging. In: Bandini, S., Manzoni, S., Umeo, H., Vizzari, G. (eds.) ACRI 2010. LNCS, vol. 6350, pp. 24–34. Springer, Heidelberg (2010)
10. Sipper, M.: Evolution of Parallel Cellular Machines. LNCS, vol. 1194. Springer, Heidelberg (1997)
11. Sipper, M., Tomassini, M.: Computation in artificially evolved, non-uniform cellular automata. Theor. Comput. Sci. 217(1), 81–98 (1999)
12. 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.E. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 432–441. Springer, Heidelberg (2007)
13. Dijkstra, J., Jessurun, J., Timmermans, H.: A multi-agent cellular automata model of pedestrian movement. In: Schreckenberg, M., Sharma, S.D. (eds.) Pedestrian and Evacuation Dynamics, pp. 173–181. Springer (2001)
14. Nagel, K., Schreckenberg, M.: A cellular automaton model for freeway traffic. J. Phys. 2, 2221–2229 (1992)
15. Désérable, D.: A Family of Cayley Graphs on the Hexavalent Grid. Discrete Applied Mathematics 93(2-3), 169–189 (1999)
16. Désérable, D.: Minimal Routing in the Triangular Grid and in a Family of Related Tori. In: Lengauer, C., Griebl, M., Gorlatch, S. (eds.) Euro-Par 1997. LNCS, vol. 1300, pp. 218–225. Springer, Heidelberg (1997)
17. Fraigniaud, P., Lazard, E.: Methods and problems of communication in usual networks. Discrete Applied Mathematics 53(1-3), 79–133 (1994)
18. Ediger, P., Hoffmann, R., Désérable, D.: Routing in the Triangular Grid with Evolved Agents. J. Cellular Automata 7(1), 47–65 (2012)