'GAME OF LIFE' WITH MODIFICATIONS: NON-REGULAR SPACE, DIFFERENT RULES AND MANY HYERARCHICAL LEVELS

Lois Facchetti, Alexander Makarenko

***Abstract**: The results of computer investigations of CA with 'game of 'Life' models, but on the cellular space with the 'whole', which have different shapes and different distribution in the space. We give results of experiments and discussed some correlations in behavior with parameters of non-homogeneities. Also the results of different rules of CA influence are presented. And finally the results for some hierarchical CA are described. Proposed results may be useful for understanding of non-classical CA behavior which may be interesting for some problems of cellular computing.*

***Keywords**: cellular automata; non-regular space; non-classical rules; hierarchical CA; computing; anticipation.*

1 Introduction

Cellular automata (CA) since past century were one of the very important fields in theoretical computer science [1-8]. Also cellular automata have many very important applications, for example, in physics [9], in crowds movement modeling [10-12], in pattern recognition [13], in brain investigations [14-16], in quantum mechanics [17, 18], in computation theory [19].

Usually all investigations focused around the most know CA – the game 'Life' by J. Conway, or some their modification, for example with probability accounting in the rules [2, 8, 11, 15]. But recently as in the theory as in the practice more essential improvement in CA arise: evolving and adaptive rules [20], asynchrony in the rules operation [21], decline from absolute homogeneities in rules for all cells [22], using of fixed similar neighbors for all cells and relies from full regularity of basic cellular space.

The letter property corresponds to common non-homogeneity of real physical spaces, problems, processes in the Nature – for example of percolation [23, 24]. CA models for such problems should correctly account such non-homogeneity. Presumable introduction non-homogeneity of basic space follows for unknown changes s in CA behavior. It posed the problems of understanding qualitative and quantitative influence of non-regularity of basic space on CA properties.

So in given paper we propose the description of some our investigation of such problem. We describe the results of our computer investigations of CA with 'game of 'Life' models, but on the cellular space with the 'whole', which have different shapes and different distribution in the space. We give results of experiments and discussed some correlations in behavior with parameters of non-homogeneities. Also the results of different rules of CA influence are presented. And finally the results for some hierarchical CA are described. Proposed results may be useful for understanding of non-classical CA behavior which may be interesting for some problems of cellular computing.

2 Description of cellular automata model on non-regular space

In classical Game of Life cellular automata, we tried to determine a rule between, on the one hand, the percentage and distribution of initial wall cells and, on the other hand, the average time steps before reaching a temporal cycle in the cellular automata.

In the numerical simulations below, we adopted the model of classical Game of Life rules, with a classical Game of Life neighborhood for the cells. Yet, we introduced a certain amount of cells considered as "walls". These walls are assigned a value at the initialization of the cellular grid. This value determines the behavior of the cells right next to a wall cell. In most cases, these walls are frozen - although in two particular cases, the walls introduced could act as classical alive cells. When the distribution of walls is set, we study the number of steps up to which the cellular automata reach a cyclic evolution. For each value taken by the walls, we draw a two dimensional graph representing the average time before cycle. It enables studying the speed for cellular automata to reach a cycle in function of the initial distribution of the walls.

2.1. Modification of 'Game of Life'

2.1.1 'Game of Life' neighborhood

We adopted the Game of Life classical model for cellular automata. It consists of a Moore neighborhood. At each step, the cell considered evolves accordingly to state of its eight surrounding cells.

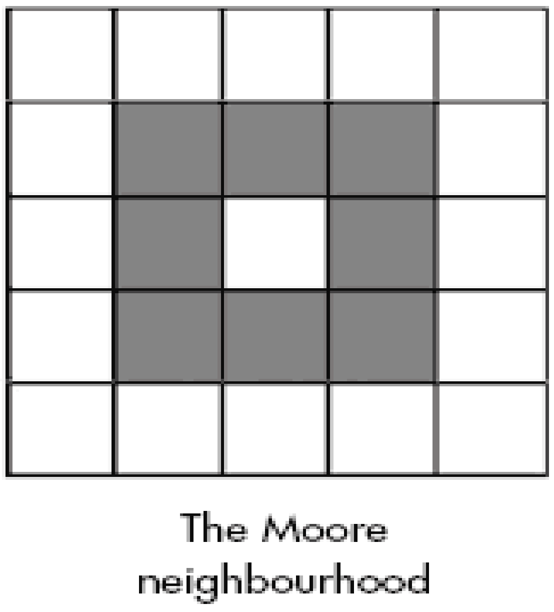


Fig. 1 Moor’s neighbor of given cell (eight gray cells)

2.1.2 ‘Game of Life’ classical rules

We adopted the Game of Life classical rule for every cell. At each time step, a cell can be in one of the two states 'dead' or 'alive' (or 0 or 1). Each cell evolves according to its own state and to its eight neighbor’s states. Here is the rule to update a cell:

- an 'alive' cell at time t-1 remains alive at time t if 2 or 3 of its neighbors are also alive. Otherwise, it becomes 'dead' at time t.
- a 'dead' cell at time t-1 becomes alive at time t if 3 of its neighbors are alive. Otherwise, it remains 'dead' at time t.

In our model, a cell is represented by a Boolean value equal to 1 if the cell is alive and 0 if the cell is dead. Hence, the rule can be rewritten as a function of the sum of its neighbor’s states. Let's consider that C(t) represents the state of one cell at time t, and NS(t) represents the neighborhood sum of values.

We can rewrite the Game of Life rule as:

- **if** $C(t-1)=1$ and $1<NS(t-1)<4$ **then** $C(t)=1$, **else** $C(t)=0$
- **if** $C(t-1)=0$ and $2<NS(t-1)<4$ **then** $C(t)=1$, **else** $C(t)=0$

Introducing strict inequalities will be necessary for the establishment of wall behavior.

2.1.3 Border conditions

The border conditions are 2D periodic, in other words our 2D cellular automata is a manifold of a 3D torus. For cellular automata composed of a NxN squared grid, considering C(i,j) the value of cell at t line i column j, we impose:

- C(N+1,k) = C(1,k) for k in {1,..,N}
- C(k,N+1) = C(k,1) for k in {1,..,N}
- C(0,k) = C(N,k) for k in {1,..,N}
- C(k,0) = C(k,N) for k in {1,..,N}

Defining a C(i,j) with i>N , j>N, i<0 , j<0 is necessary for the definition of the neighborhood of border cells.

2.1.4 Initial distribution of cell's states

For the numerical simulations, we will consider that the initial available grid -i.e. the whole grid deprived with the "wall" cells- is filled with uniformly-distributed fifty percents of alive cells.

2.1.5 Introduction of the walls into homogeneous cellular space

Now, we need to define the "walls" of our model (or distribution of the wholes).

A rule for the cells near the 'walls'.

Here we describe one of the possible rules for evolution for the cells near the walls. Remark that applications the rules for the cells near walls may correspond to physical nature of considered problems. Contrarily to normal evolving cells, "wall" cells can take any real value. This hypothesis describes entirely the model adopted, without having to change the previously enunciated rules. Keeping the previously introduced notations, the rules remain:

- **if C(t-1)=1 and 1<NS(t-1)<4 then C(t)=1, else C(t)=0**
- **if** C(t-1)=0 and 2<NS(t-1)<4 **then** C(t)=1, **else** C(t)=0

Both of the previous conditions are still implementable whether the values in the sum of neighbors are different from 0 or 1.

We could add the explicit condition: if the cell is a wall, then the cell remains a wall. Using, the previous notations:

– if C(t-1) ≠ 0 and C(t-1) ≠ 1, then C(t) = C(t-1)

It appears that this ultimate condition is optional. By setting such values to wall cells, the behaviors of the cells near to walls cells are changed. These local changes affect the whole cellular automata. Note that without changing the above conditions, if the walls have the value 0 or 1 at initial time step, they behave as normal cells in the following step.

2.1.6 Description of distribution and of wall's shape

In our simulation we for simplicity consider that the walls are square-shaped group of cells. Before each cellular automata simulation, we define a size of square and a percent of wall cells among the whole grid. The variable *s* represents the size of the square of surface containing s^2 wall cells. The percent represents the total amount of wall cells divided by the total number of cells in the grid.

At initialization time, the grid is full of empty cells. In given investigation we add successively one randomly placed *s*-sized square to the grid, till the percent of walls cells initially defined is exceeded. The effective percentage (in general different from the intended one) is eventually computed.

Here are presented the 6 first steps leading to the construction of the wall cells. The effective percent of wall cells is indicated. The intended percentage was 50%, the size of square is 10. The size of the grid 30x30.

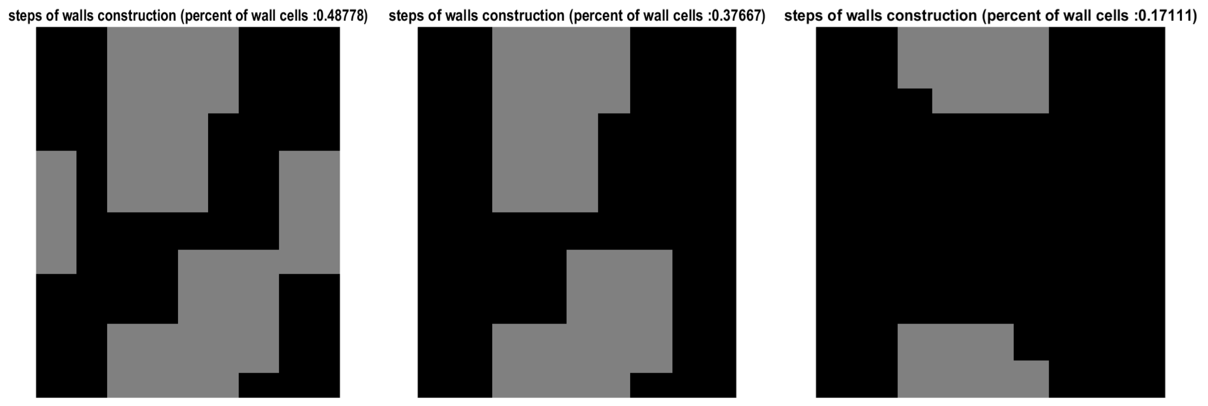

steps of walls construction (percent of wall cells :0.48778)
steps of walls construction (percent of wall cells :0.37667)
steps of walls construction (percent of wall cells :0.17111)

Fig. 2 Steps of wall construction

2.2. Numerical simulations

All the simulations were realized using Matlab software. In this section, we present the model, the parameters set and some results of the experiments.

2.3 Computing model

The cellular automata were modeled using matrix of size N^2xT, with **N** representing the side of the square grid, and *T* the maximal number of time step. The cell can take real values, 0 for dead cells, 1 for alive cells and arbitrary real number for walls different from 0 and 1. Each N^2 submatrix represents the state of the cellular automata at a particular time. The submatrix of the cellular automata at time t is represented by the cells of coordinates (i,j,t), with i,j in {1,...,N}, and t in T. By analogy with Matlab representation, we will note this matrix M(:,:,t).

Initialization During the initialization step, M(:,:,1) is randomly and equally filled with 0 and 1. Next, the wall cells are created according to the process presented in the model presentation. The wall cells values replaces the values of the cells in M(:,:,1) matrix. At this point, M(:,:,1) is filled with at most three different values : 0 (dead cells), 1 (alive cells), an arbitrary value (wall cells).

Update At each step, each cell of M(:,:,t) is updated according to its neighbors at time t-1, according to the Game of Life rules.

End The cellular automata can stop evolving for two reasons:

- the cellular automata reached the final time step $t=T$. T is returned.
- The cellular automata reached a cycle state. The time t at which the cyclic state was detected and the length l of the cycle is returned.

2.4 Parameters in the model

In this section, we will define the parameters used for the numerical computation.

a. size and space of the cellular automata

We used a 30x30x500 Matrix to model our cellular automata, i.e. a grid of 30x30 and a maximal time step of 500. These measures offer a good tradeoff between complexity of the evolution and computation speed. In the majority of the cases, the cellular automata reached a cyclic state before 500 iterations step, which is necessary to truncate as few as possible the cellular automata evolution, which could perturb the data.

b. detect cycles in the model

We enable the algorithm to detect a cycle of at maximum 12 time step. In practical experiments, this number seems appropriate as most of cycle of length 8 and above is very rare, yet present.

2.5 Evolution of one CA over time

Here, we present the visual representation of a cellular automata evolving along the time. Some shots are represented, including the initial Matrix and the final one (i.e. the one at which we detected a cycle).

Black points represent dead cells, white one alive cell, and grey on wall cells. Initial intended percent of wall is 0.40.

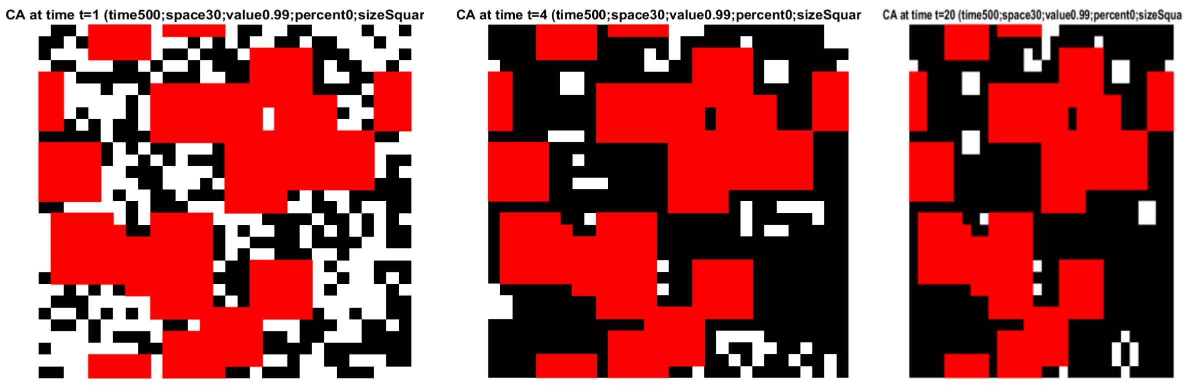

CA at time t=1 (time500;space30;value0.99;percent0;sizeSquar
CA at time t=4 (time500;space30;value0.99;percent0;sizeSquar
CA at time t=20 (time500;space30;value0.99;percent0;sizeSqua

Fig. 3 Example of CA evolution during time

2.6 Parameters in the experiments

After having seen evolution of such cellular automata along time, we now intend to draw a rule binding the time to reach cycle and the initial distribution of walls. Therefore, we will draw a plot representing the average length time to reach cycle, in function of two parameters: size of square walls and percent of wall cells. To do this, we will represent the length time with the intensity (black = 0, white = max) on an image with coordinates x representing the size of wall squares, coordinates y representing the percent of wall cells.

2.6.1 Number of simulation and values of the walls

We simulated cellular automata for eleven different values for the wall. These values have been chosen to observe behavior as diverse as possible. Here are the values studied: {-1, -0.01, 0, 0.01, 0.5, 0.90, 1, 1.01, 1.5, 1.99, 2}.

For each of these peculiar values, the simulation computed whether 20,000 or 40,000 different evolution of cellular automata, depending on the time the simulation was taking. The simulation returned as many values for *t* -the time before reaching cycle- and l -the length of the cycle.

2.6.2 Percent and size of wall squares

For every value of the walls, we obtained 20,000 or 40,000 couple of values (*t, l*) which were obtained by setting different parameters:

- *p* (percent of wall cells) : going from 0.001 to 1.000.
- *s* (size of the square walls) : integer values from 1 to *sqrt(p)*N*.

There are two nested loops: the first one runs on *p*, the second one runs on *s*. For a fixed p, we simulate all the integer size of square s, satisfying the condition: $s^2/N^2 < p$. Note that here, *p* is the intended percent, not the effective one. We need to return the effective percent p_e of walls at initial time.

When we obtained the 20,000 (or 40,000) values for the value wall (*t, l*) in function of (p_e, *s*), we classified them according to their length cycle l.

Next, we averaged them by grouping the value which had the same size s and the same integer part of effective percent. For example, if one value was computed with initial wall parameter *s*=2, p_e=10.68 and another for *s*=2, p_e=10.55, then they are averaged together.

At this point, for each value of walls, we have matrices M_l of values $M_l(p, s)$ with p in {1,...,100} and s in {1,...,N} representing the average time for cellular automata to reach cycle for *p* percent and *s* size of walls. Once these matrix M_l are obtained, we need to visualize them.

2.6.3 Representation

For each value of walls and for each cycle, the matrices have been visualized thanks to gray scaled images, where the intensity of the pixels represents the relative average time steps. "Relative" means that for each Matrix, we extracted the highest time step value t_m and divided every time step value t by this value, to obtain a ratio r going from 0 to 1, representing the relative average time steps. In order to grant highest values a sharper importance, we also applied a function to these [0,1] values to get r_f value: $r_f = r^{3/4}$. Finally, on an image of size 100x100, we represent the r_f intensities.

2.7 Results

In this section we present and explain some of the images simulated. Here is the examples image representing the 30x30 cellular automata for walls value of 0.01, ending by a cycle of length 1. We can read that the maximum average time for these automata is 480. It is the white point on the image, reached for percent around 75% and size around 70). Note that percent is 0 at the top and 1 at the bottom of the image, while size of square is 0 on the left and 100 on the right. Here the cycle "-1" means that these cellular automata did not reach a cycle within the T=500 time steps. Here the global-cycle means all the cycles of the walls value 0.99 have been represented on the same graph.

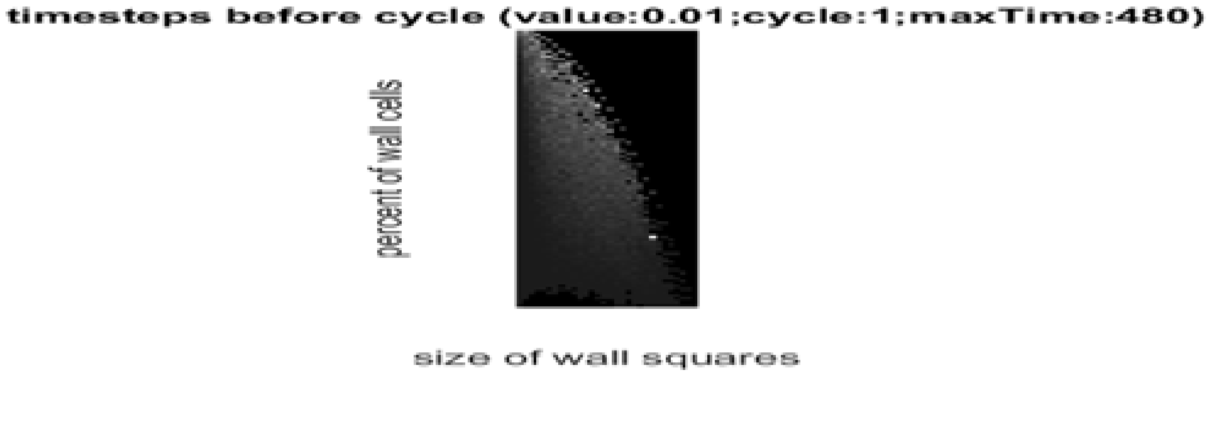


Fig. 4 Cycle

2.8 Analysis of the curves

In this section, we briefly discuss the global trend occurring in the graphs.

2.8.1 Global cycling

This subsection observes the global behaviors of the cellular automata without pointing out at precise cycle length, but taking all of them into account. Three main ideas can be noted.

First, there seems to be a global common distribution of average time steps before cycle stabilization. With a higher time step for small percent value of walls and small values of wall squares size. This image seems to be the archetype example of the typical global distribution.

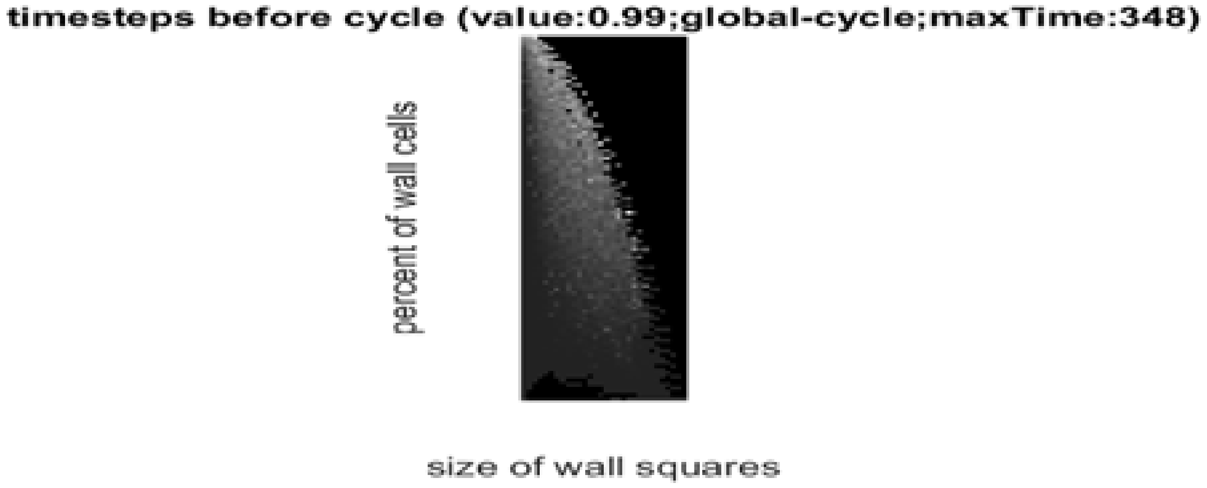


Fig. 5 Global cycling under value 0.99

Second, as we could expect, the behaviors observed for walls values of 0 and 1 and sharply different from the other ones. The average time step is far more homogeneous over the graph, and the average time step are higher than those for the other rules. Let's remind that these are not frozen wall cells as it is the case for the other walls values, but acts like evolving cells from time step 2. We can notice there is no continuity in function of wall values around 0 and 1. Lastly, some values such as 0.5 and 1.5 makes appear a high time step probability for values with intermediate percent and low square size.

2.8.2 Behavior in dependence of cycle length

In this section, we try to discriminate behavior in function of cycle length, at fixed values. First and foremost, what can be noticed is that most cellular automata end with a cycle of length 1 or 2. Their distribution is very similar. Next, automaton with cycle length is 3, 4 and 6, and those with no detected cycle in 500 steps, are the most frequent. Here we present examples of highly represented cycles.

Conclusion to Section 2

We were able to determine to some behaviors influenced by parameters of the wall distribution. We try to evolve in a context as constrained as possible (with square walls) to avoid coping with a huge number of parameter, which could have been overwhelming in the visualization step. We could see some rules appearing for global cycle, such as a higher average time step before cycle for low percentage and high size of wall squares. Some of the data found were presented in this report, but most of the data are still to be exploited from all the other data.

3. 2D Cellular automata: Study of Cellular Automata with inhomogeneous neighborhood and non-classical rules

3.1 Problem

We report a series of numerical experiments carried out on 2D cellular automata. We studied the behavior of cellular automata with different activation rules and with different inhomogeneous neighborhood. Rules implemented are inspired from classical rules of Conway's Game of Life. Neighborhood is a random function which takes in account user-set features. This series of experiments is not an exhaustive study of the whole problem. It intended to draw conclusion in a voluntarily constrained framework to obtain readable results. The whole study is empirically tackled.

3.2 General Model

In this section, we present the global model adopted for the simulation of the cellular automata. Each experiment has its own parameters and programming specificities. We won't present any of these specific sets. Instead, we introduce the general iteration process of the cellular automata. The Conway's Game of Life classical rules had been described in the Section 2.

3.2.1 Generalized rules implemented

The rules implemented are based on the classical Game of Life rule idea. Here, we suppose that NS(t) represents the sum of the neighbor's value of the considered cell at t. Our rules can be modeled as a vector of four integer values (a, b, c, d). With the same notations that previously, the rule can be rewritten as:

- for C(t-1)=1 : if **a < NS(t-1) < b** then C(t)=1, else C(t)=0
- for C(t-1)=0 : if **c < NS(t-1) < d** then C(t)=1, else C(t)=0

For instance, classical Game of Life rule corresponds to the vector (1, 4, 2, 4). Note that we should have a < b-1 in order to enable cells being maintained alive, and have c < d-1 in order to enable them recover from death.

3.2.2 Neighborhood

To build the neighborhood, we made an analogy with graph theory. The neighborhood of our model is inhomogeneous, directed, and constant over time.

Initially, we attribute two parameters to the neighborhood of the grid:

- the number of neighbor's n of each cell.
- the radius r of the neighborhood, which can be seen as the integer value of the neighborhood scope. If we consider the cell at coordinate (i,j), another cell (k,l) can be in its neighborhood if:

r <= max(|i-k|, |j-l|)

Here is a figure illustrating the neighborhood

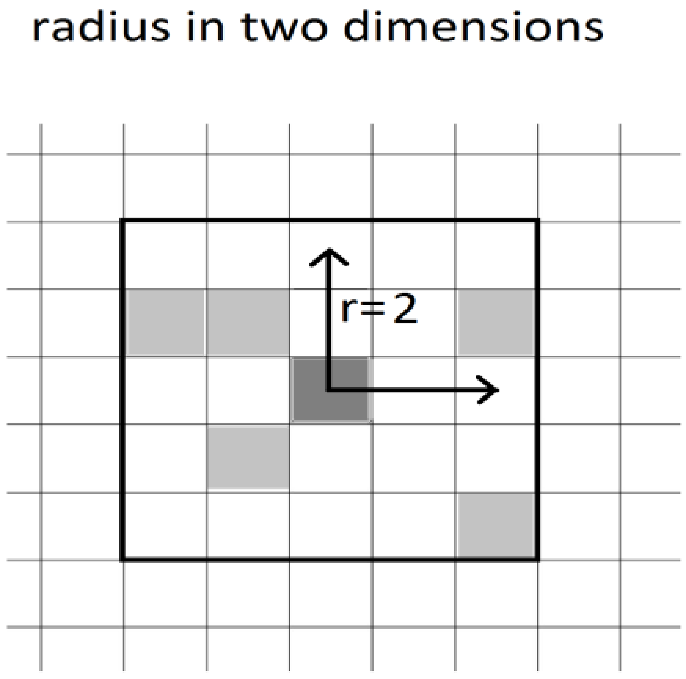


Fig. 6 Generalized neighbor and rules

Dark grey cell represents the cell we consider. Light grey cells are the neighbors of the cell. They are randomly chosen among the cells in the r-radius square.

At initialization step, each cell is given n neighbors among the $(2r+1)^2-1$ cells which subscribe the r radius condition. The choice of the cells is uniformly random. We also impose that cell cannot be their own neighbors. At updating time t-1→t, each cell computes **NS(t-1)** – the sum of its neighbor's value – and evolves in accordance with this value, and with its own state.

3.2.3 Border conditions

The border conditions are 2D periodic, in other words our 2D cellular automata is a manifold of a 3D torus. For a 2D cellular automaton composed of a NxN squared grid, considering C(i,j) the value of cell at t line i column j, we impose:

— C(N+m, k) = C(m, k) for k, m in {1,.., N}

— C(k, N+m) = C(k, m) for k, m in {1,.., N}

— C(1-m, k) = C(N-m, k) for k, m in {1,.., N}

— C(k, 1-m) = C(k, N-m) for k , m in {1,.., N}

Defining a C(i, j) with i>N , j>N, i<1 , j<1 is necessary for the definition of the neighborhood of border cells.

3.2.4 Initial distribution of life

At initial time step, every cell is given a fifty percents probability to be alive. Other cells are dead.

3.3 Numerical experiments

In this section, we present the series of experiments we conducted. The general aim is to study the overall evolution of the cellular automata. Hence, we used some global representation plot: evolution of the cellular automata over time, life rate of the cellular automata over time, average time steps to reach cyclic state in function of space.

For all the numerical experiments, we used Matlab software to simulate the cellular automata. The maximal duration – number of time steps simulated – of cellular automata was 5000. This is a convenient value in term of computation time. We used a grid of 30x30 cells.

Here, we present the average number of time steps required to reach a global cycle in function of the size of the grid. A global cycle is defined as a cycle in the temporal evolution of the whole grid. Here, this simulation indicates that we modeled cellular automata from size 3x3 to size 35x35, and we did this three times ("nbValues3" in the title of the graph). Then we averaged them for each size. For instance, the 30x30 grid reached a global cycle after 600 time steps in average. This preliminary experiment was useful to test whether the 30x30 size chosen was enough to enable the cellular automata to converge in a global cycle within less than 5000 time steps.

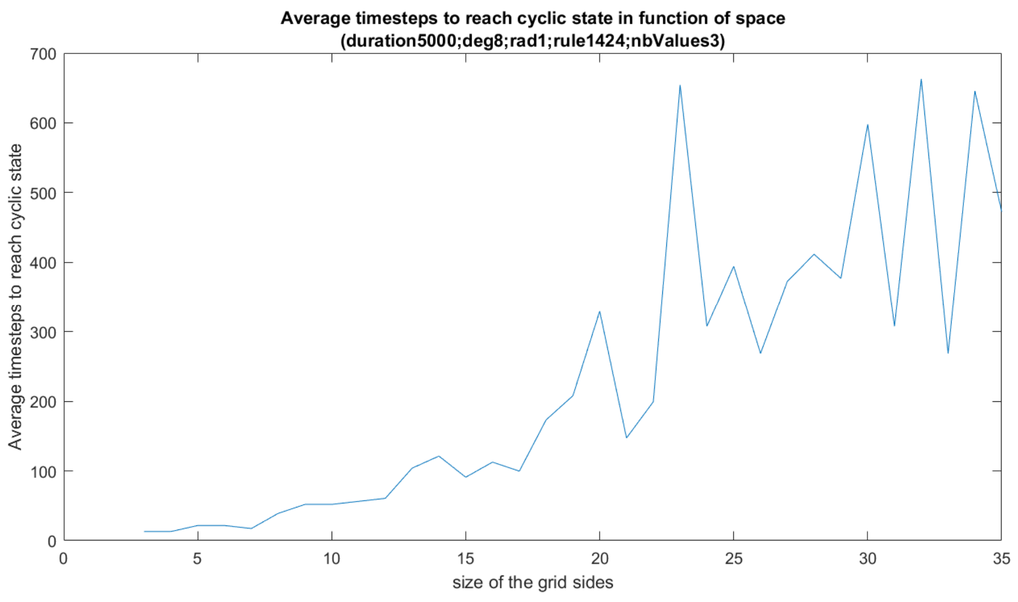


Fig. 6 Time of reaching cycles

3.3.1 Life rate over time in function of number of neighbors

In these first simulations, the aim was to detect particular comportments of the cellular automata in function of the number of neighbors we set. All the simulations consider the classical Conway's Game of Life rule – which we named (1, 4, 2, 4) in our model.

On each graph, we represented the percent of alive cells of the cellular automata over time. As we explained before, each graph represents different cellular automata: different initial distribution of life, and different neighborhood for each cells.

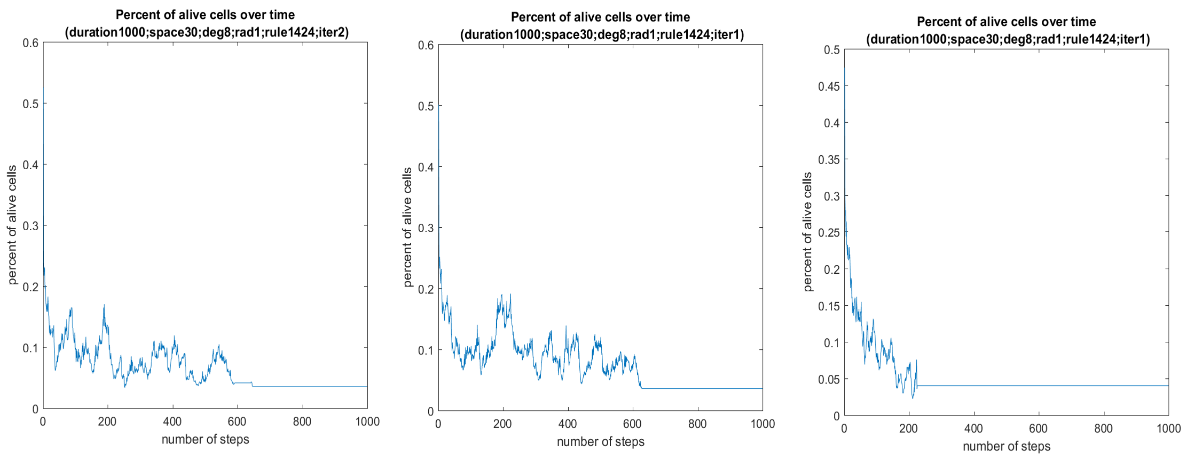


Fig. 7 Simulations for 8 neighbors (classical Game of Life).

As displayed on these nine different graphs, most cellular automata converge within a number of time steps inferior to 1000. What's more, when the number of neighbors decreases the number of time steps necessary to reach a global cyclic state decreases. Besides, the final rate of alive cells also seems to decrease with the number of neighbors.

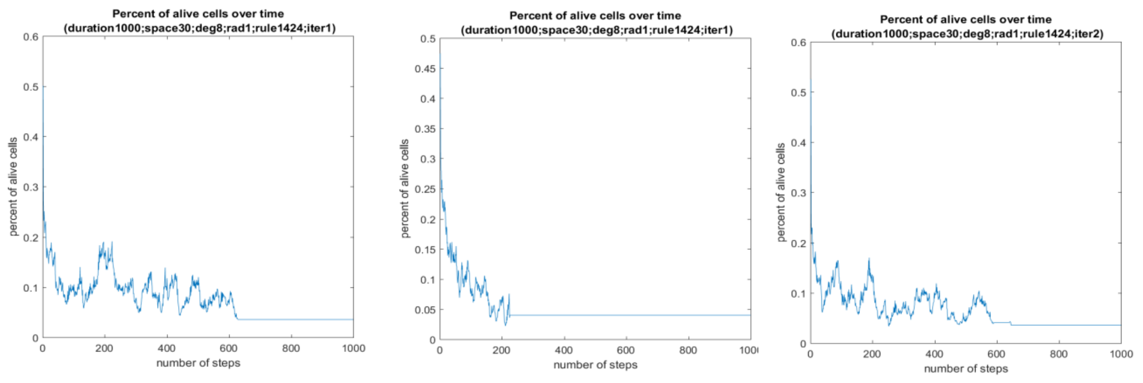
This last point can be must be due to the sum of the neighbors' states NS(t) whish drives the rule and which is lower when diminishing the number of neighbors. Hence, the condition 1<NS(t)<4 which enables a cell to remain alive and the condition 2<NS(t)<4 which enables a cell to get alive back is statistically harder to reach. Hence this fall of the life values over time.

3.3.2 Life rate over time in function of the rule

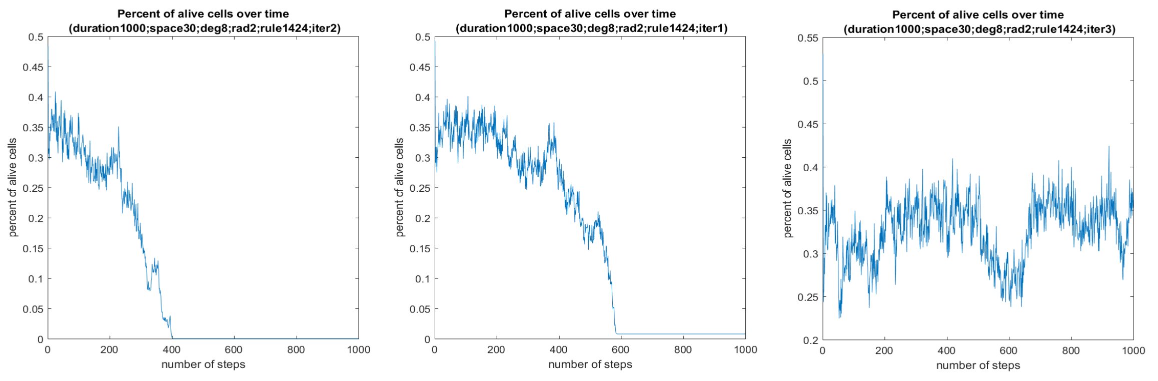
In the next simulations, the aim was to detect particular comportments of the cellular automata in function of the radius set. All the simulations consider the classical (1, 4, 2, 4) rule.

On each graph, we represented the percent of alive cells of the cellular automata over time.

Simulations with radius 1 (classical Game of Life):



Simulations with radius 2:



Simulations with radius 3 :

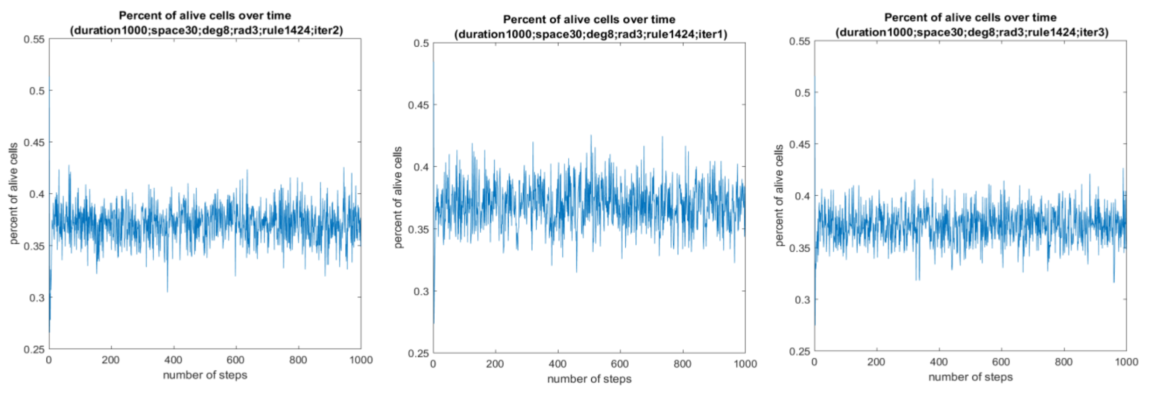


Fig. 8 Simulations with different size of neighbors

As displayed on these nine different graphs, when the radius increases the number of time steps necessary to reach a global cyclic state increases. Plus, the fast oscillations of the life rate increases as the radius grows. These oscillations of life seem to have an increasing amplitude and frequency. Besides, although the oscillations increase, the general trend of life over time seems more stable easier to forecast. We also notice that for a radius 2, the cellular automata converging in a cyclic state often converge in a global dead state. Unfortunately the maximum duration time steps does not enable us to draw the conclusion for radius=3.

3.3.3 Average time steps to reach cycle in function of number of neighbors

In order to draw more accurate figures on cellular automata – especially regarding the number of timestamps needed to reach a global cycle – we conducted other simulations. In the following simulation, we once again used the classical (1, 4, 2, 4) Conway's Game of Life Rule). The goal of the simulation was to study the average number of time steps before reaching a global cycle in function of the number of neighbors, going from 2 to 8. For each number of neighbors, we computed 41 different cellular automata and averaged their number of time steps required to reach a global cycle:

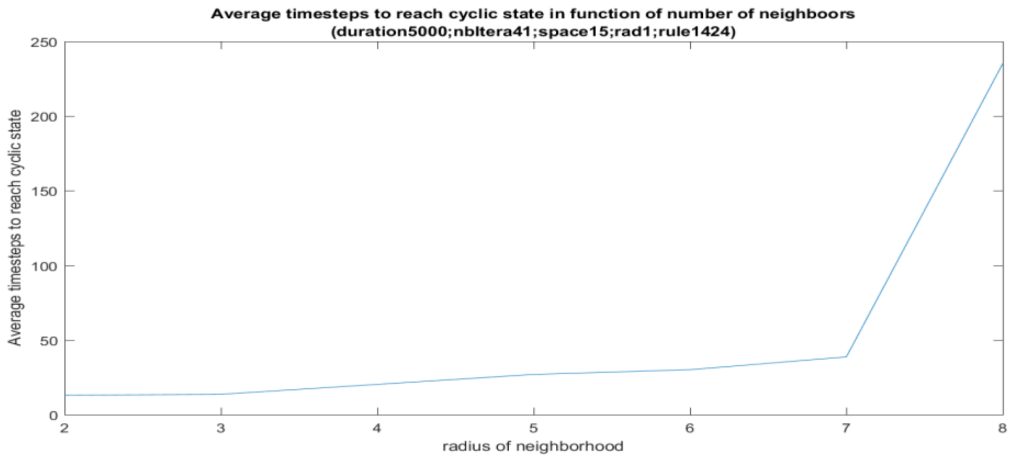


Fig. 9 Averag time for reaching cycles

The graph shows a very clear break for the critical value 8 of classical Game of Life rule. For value from 2 to 7, the average number of time steps looks like an increasing linear function of the number of neighbors. Note that it was worthless simulating the case with one neighbor since, following the Gale of Life rule, every cell would have died within the first simulated time step.

3.3.4 Average time steps to reach cycle in function of radius

Here, we represented the same type of graph, but changing the neighborhood scope, i.e. increasing the radius parameter.

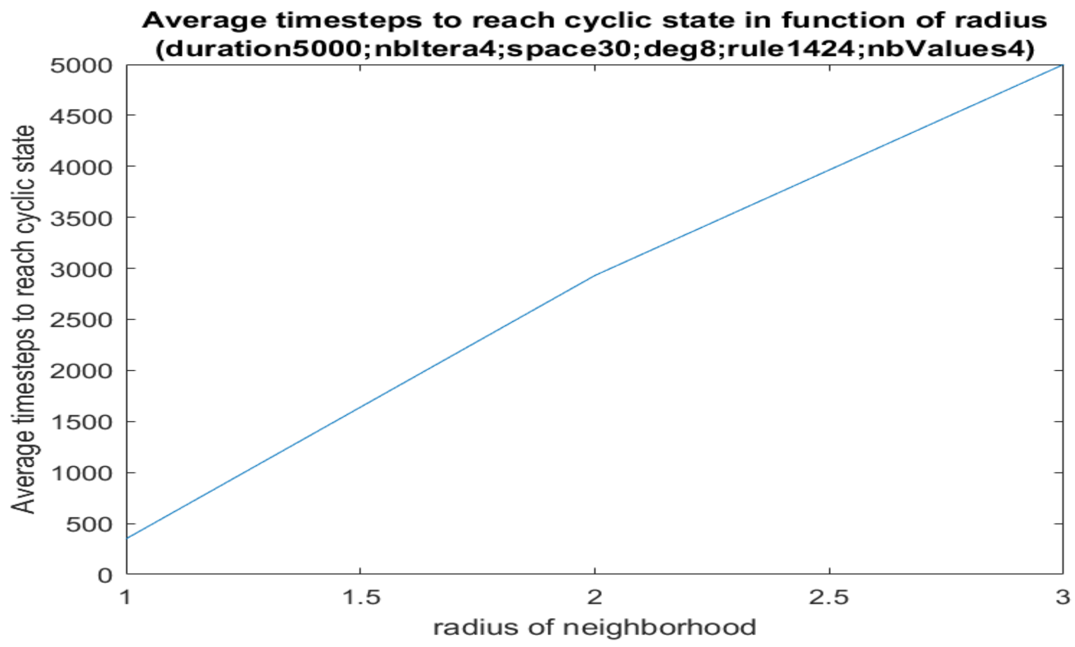


Fig.10 Average time for cycling in dependence on radius.

Here, the conclusion of the graph is limited by the maximum number of time steps we imposed on our cellular automata. Above a radius of 3, the average time steps overcome the 5000 time steps maximal duration. Yet, we have data about the radius 2 neighborhood's average time steps to reach global cycle, which is 6 times higher than the one for the radius 1 neighborhoods.

3.3.5 Determination of Game of Life-alike rule with different neighborhood

Some previous graph pointed out the limit of the study we conducted: the radius and the number of neighbors of each cell are closely linked to the classical Game of Life rule. Hence, changing one of these two parameters leads to sharply different behaviors. This emphasizes the link between neighborhood and rule in cellular automata behavior. This last experiment is a short empirical study aiming at determining some Game of Life-alike rules which lead to Game of Life-alike behaviors.

In order to determine such rules, we implemented algorithms based on random and mutation. Such algorithms are useful to determine "good" rule without examining each rule, due to the lack of computational power. Here, we do not have any objective function as it could be the case in genetic algorithm. Instead, we test the final life rate of the cellular automaton and if it belongs to the expected range, the rules are selected and then undergo some mutations. The conditions upon life attempt to find a life rate close to the 7% that the classical Conway's Game of Life converges to.

Here are the different steps of the algorithm:

Stage 0 : Fix a number of neighbors and a radius for the neighborhood.

Stage 1: Generate random (a,b,c,d) and select them if the cellular automaton state agrees with "loose" life conditions – life rate at time step 100 between 0.01 and 0.4.

Stage 2 : Make evolve randomly every (a,b,c,d) already selected at stage 1 and simulate the corresponding CA. Select the new rules if the cellular automata state agrees with "medium" life conditions – life rate at time step 200 between 0.01 and 0.4.

Stage 3 : Simulate CA with (a,b,c,d) already selected at stage 2 and select the rules if the CA agrees with tight life conditions – life rate at time step 1000 between 0.02 and 0.3.

We show the evolution of CA whose rule has been selected by the algorithm.

The cellular automata found with this rule enable a certain kind of life. The emergence of such life pattern is not trivial and most rules generated randomly do not enable finding such shapes. Most of the time they converge to a dead state.

This last experiment is an attempt to highlight the link between the neighborhood and the rules implemented in the evolution of life in such systems.

Conclusion to Section 3

Here we report the study of 2D cellular automata with inhomogeneous neighborhood and with new types of rules, inspired from classical Game of Life cellular automata. We built a model for the neighborhood, making analogies with network science through graph theory. We also defined a new model for the rules, and explained how we could transpose this model for Game of Life classical rule, and how we could derive new rules from this classical rule. Then, we simulated some cellular automata with classical Game of Life rule. We could highlight some behaviours of the life in the cellular automata. Eventually, we admit that instead of rewriting neighborhood of cells independantly of rule, which lead to significantly different cases from the classical Game of Life, we should attempt to discover new rules, more closely bound to the neighborhood adopted. They lead to some interesting self organizing pattern.

Some further studies could be carried out to extend the first graphs drawn, in particular those on which the maximal timestep was limiting the scope of the results. One might also consider other ways to discover the rule we shed light on, and thanks to more powerful computationnal means, try to investigate these rules in a more exhaustive way.

4. Study on 2D cellular automata: Multi-layer cellular automata, with classical Game of Life Rules

4.1 Problem

This report intends to study the behavior of two-layer cellular automata. Each layer of the cellular automata updates according to classical 2D Game of Life model. Some inter-layer links are introduced in the model. We studied the influence of such links upon the evolution of life in the cellular automaton.

4.2 General Model

In this section, we present the global model adopted for the simulation of the cellular automata. We make an analogy between network science lexicon and cellular automata one. In particular, an "edge" or "link" between two cells of cellular automata represents a relation of neighborhoods between these cells. In cellular automata lexicon, such cells (bound by an edge/link) are called "neighbors".

4.2.1 Multilayer model

Our model consists of two grids of size NxN. Each of these grids is a layer of the cellular automaton. These layers are connected by some inter-layer links. These inter-layer links are undirected edges. We only consider one type of inter-layer link: coupling edges. They are links connecting a cell A at a position (i,j) on the first grid, and a cell B with the same coordinate (i,j) on the other grid.

4.2.2 Updating rule

Here we remember classical Game of Life rule, which we applied to our model. At each time step, a cell can be in one of the two states 'dead' or 'alive'. Each cell evolves according to its own state and to its neighborhood's state – we will define its “neighborhood” later. Here is the rule to update a cell:

- an 'alive' cell at time t-1 remains alive at time t if 2 or 3 of its neighbors are also alive. Otherwise, it becomes 'dead' at time t.
- a 'dead' cell at time t-1 becomes alive at time t if 3 of its neighbors are alive. Otherwise, it remains 'dead' at time t.

In our model, a cell is represented by a Boolean value equal to 1 if the cell is alive and 0 if the cell is dead. Hence, the rule can be rewritten as a function of the sum of its neighbor’s states. Let's consider C(t) represents the state of one cell at time t, and NS(t) represents the neighborhood sum of states. We can rewrite the Game of Life rule as:

- in case C(t-1)=1 : if **1<NS(t-1)<4** then C(t)=1, else C(t)=0
- in case C(t-1)=0 : if **2<NS(t-1)<4** then C(t)=1, else C(t)=0

4.2.3 Neighborhood

Our model neighborhood consists of two types of neighbors:

- *intra-layer neighbors*. They represent the neighbors of the cell within the same layer. These neighbors follow the classical Moore neighborhood, i.e. they are the eight closest cells of the considered cell.
- *inter-layer neighbors*. We only consider inter-layer neighbors bound by coupling edges, i.e. cells with same coordinates but different layers.

Both type of neighbors - inter-layer and intra-layer - are considered equally regarding the updating rule. For a cell, the neighborhood sum at time t takes in accounts both intra-layer and inter-layer neighbors.

4.2.4 Border conditions

The border conditions of each layer are 2D-periodic, which is tantamount to saying that each 2D cellular automata layer is a manifold of a 3D torus.

We define the 2D-periodic condition. Every cellular automata layer composed of a NxN squared grid, considering C(i,j) the value of cell at t line i column j, we impose:

- C(N+m,k) = C(m,k) for k,m in {1,..,N}
- C(k,N+m) = C(k,m) for k,m in {1,..,N}
- C(1-m,k) = C(N-m,k) for k,m in {1,..,N}
- C(k,1-m) = C(k,N-m) for k ,m in {1,..,N}

Defining C(i,j) with i>N , j>N, i<1 , j<1 is necessary for the definition of the neighborhood of border cells.

4.2.5 Initial distribution of life

At initial time step and for each layer, every cell is given a fifty percents probability to be alive. Other cells are dead.

4.3 Numerical experiments and results

We realised all the following experiments with Matlab Software, using matrix as representation of the cells.

4.3.1 Parameters

For the numerical simulations, we adopted a two-layer cellular automata, with a size of 30x30. The cellular automata was encoded on a 30x30x2 Matrix. Each cell is assigned a boolean value in function of its state (1 for alive, 0 for dead). The last coordinate (of size 2) represents the layer of the cell. In all experiments, we simulated the cellular automata during 500 timesteps. We will consider that "final" state of the cellular automata is the state at the 500th step.

4.3.2 Life rate over time

In the next simulations, we intended to draw general a trend of the evolution of life – final life rate, time steps before reaching a global cycle – for different percents of inter-layer edges. First, we plotted the evolution of life rate – percentage of alive cells among all the grid's cells – over time for several values of p. The model is equivalent from one layer to another. Hence, to simplify the following graphs, we only displayed the rate of one of the two layers. On each plot, we displayed 10 different curves. They represent the life rates over time of 10 different cellular automata, changing slightly the initial inter-layer link probability p. On the following graph, we represented for example the life rates of 10 cellular automata with p between 0.005% and 1.000%.

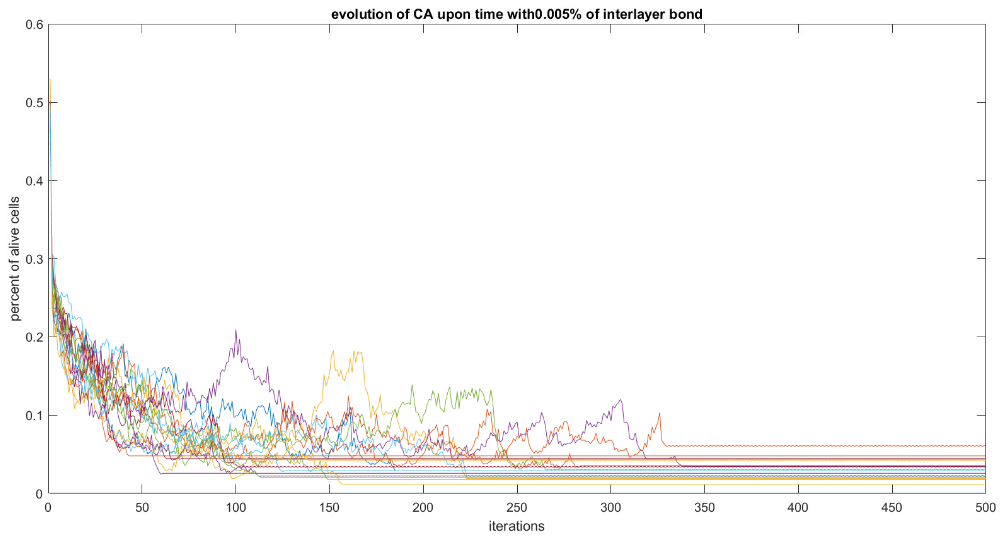


Fig. 11 Evolution of cells state

From analysis of such computations, it seems that p – the rate of inter-layer links – has an impact on the evolution of life. More accurately, it seems that:

- When p increases, the final rate of alive cells first decreases on average (until p=0.60%), before increasing.
- When p increases, the number of time steps required before the cellular automata enters a global cyclic state decreases.
- When p increases, the cellular automata tends to finish in a two-steps cycle, until p is high (around p=0.8%), where the cellular automata ends in a one-step cycle.

With the numerical experiments, we will try to confirm these predictions, and draw more precise laws.

4.3.3 Final life rate in function of interlayer link probability

In these numerical simulations, we computed the final life rate. We computed this final rate in function of the inter-layer link probability. For the simulations, p goes from 0.005% to 1.000% by steps of 0.005%. We have 200 different values. In this first graph we represent each couple (final life rate; inter-layer link probability).

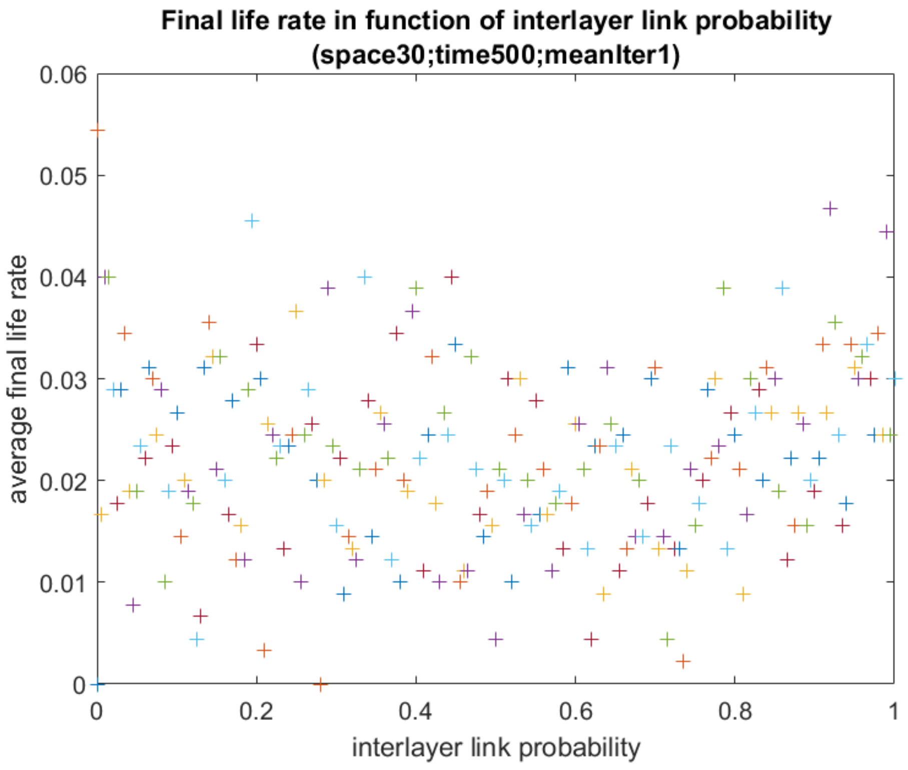


Fig. 12 Dependence on the links between layers

In this second graphs, we computed the bary-center of the previously computed couples. There are 20 bary-centers, each averaging 10 closest couples (final life rate; inter-layer link probability). Statistically, there is first a decrease of the final rate, going from around 0.030% for low p to 0.020 for p around 0.6%, followed by an increase which reaches back 0.030% for high values of p.

4.3.4 Time steps to reach cycle

We consider that the cellular automata reaches a *(global) cycle* when it is in a state it has already been in since the beginning of the simulation. A state of the cellular automata represents the state of every cell on both layers. Here, we implemented a basic algorithm which compares the state of the cellular automata at time t with all the previous states it was in. In these numerical simulations, we compute the time steps when the cellular automata are detected to be in a cycle. We computed these time steps in function of the inter-layer link probability. For the experiments, the p probability varies from 0.005% to 1.000% by 0.005%. Hence, we had 200 different values. We also had computed the bary-center of the previously computed coupled. There are 20 bary-centers, each averaging 10 closest couples (time steps to reach cycle; inter-layer link probability). From the computations it is clear that there is a relationship between the inter-layer link probability and the number of time steps to reach a cycle.

As the density of couples (time steps to reach cycle; interlayer) is low for p<0.2, we also simulated another 200 values for p between 0.001% and 0.200%. As we can see the number of time steps required to reach cycle is higher for low p. Hence, exceptionally, this simulation computed up to 1000 time steps for each cellular automaton. The graphs of computation to confirm the trend highlighted before.

4.3.5 Cycle length in function of inter-layer link probability

Here, we plotted the length of the cycle (within a cycle, minimum number of timesteps before reaching twice the same state). Some cellular automata did not reach a cycle within the 500 timesteps. They are given the length cycle 0 on the graphs. We simulated the cellular automata for the same values of p as previously.

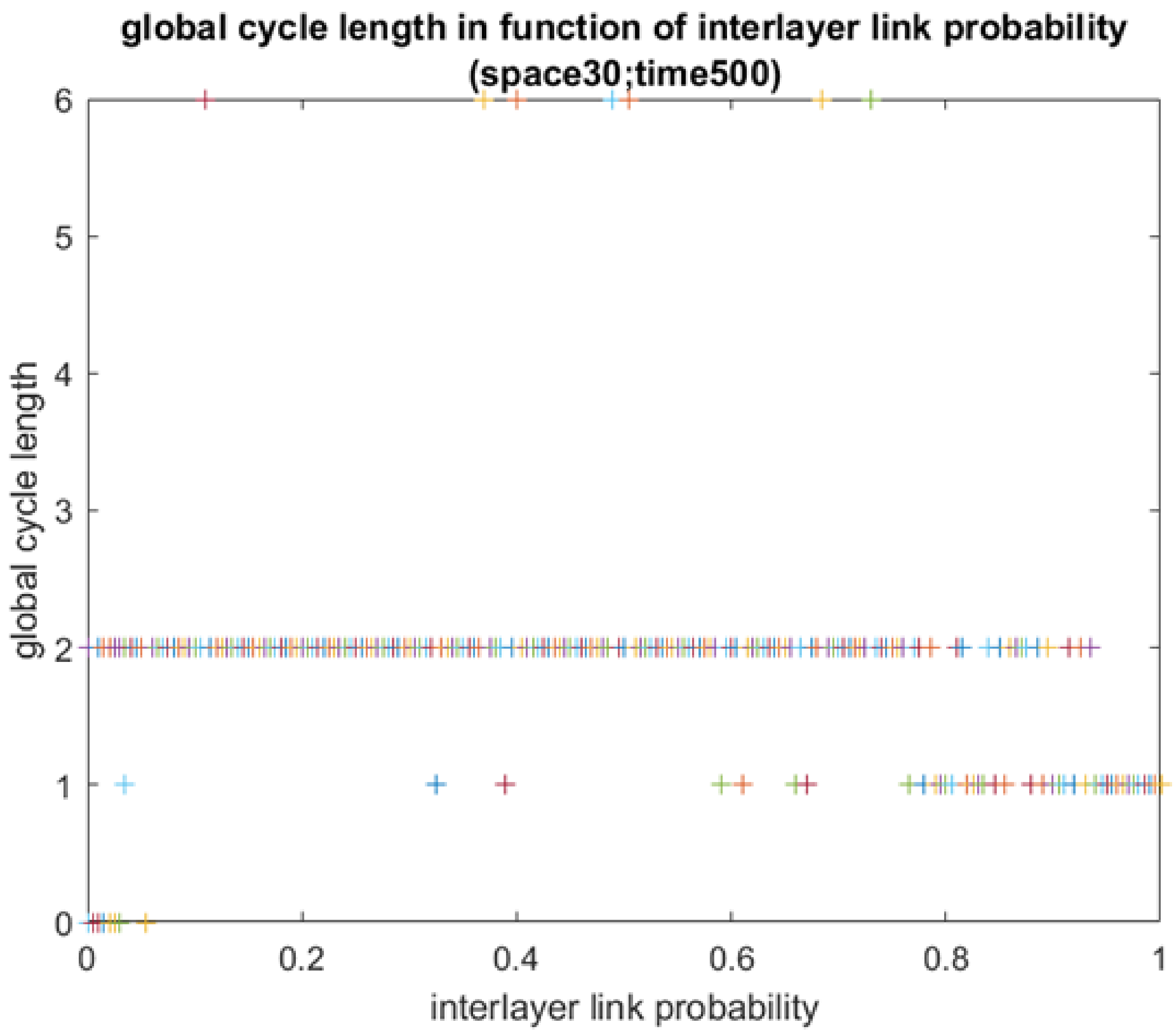


Fig. 13 The cycle length dependence of interlayer coupling.

The graphs enable us to draw a statistical trend of the cellular automata, which are highly prone to:

- end in a two-state global cycle for p<0.80%.
- end in a one-state or two-state cycle for 0.8%< p <0.93%.
- end in a one-state cycle for p>0.93%.

Conclusions for Section 4

This short study on multi-layer cellular automata give some general trends of life evolution in a constrained framework of cellular automata. We manage to draw statistical laws bewteen the inter-layer link probability on the one hand, and the number of timesteps to reach cycle, the final life rate, and the cycle length on the other hand. We decided to restrict the study to two-layer cellular automata with exclusively coupling edges. This choice was imposed by the important number of parameters already existing, the presence of random conditions which might make the trend harder to draw. Besides, adding another layer would add the number of neighboors, which could affect the influence of classical game of life rule. It would also require to reflect on inter-layer linking. With some linking conditions, each layer might be no longer symmetric (and equivalent) from the system. Adding new layers, or introducing other types of inter-layer links might motivate some other studies.

General conclusions.

Thus in given paper we described the results of investigations by computer experiments with different kinds of the classical cellular automata generalization. Received results may be useful for understanding the behavior of cellular automata which may be useful for searching the architecture for cellular computing. Remark that deep further generalizations should be considered. The first is related with the accounting of networks structure especially the ‘small world’ property. The second prospect for cellular computing is accounting of strong anticipatory property (see [1, 16, 18]).

Acknowledgement

“The paper is published with partial support by the ITHEA ISS and the ADUIS”.

References

1. *Makarenko A*. Cellular Automata with anticipation: Some new Research Problems // Int. Journal of Computing Anticipatory Systems (Belgium). — 2008. — **20**. P. 230 – 242.
2. *Von Neumann J.* Theory of Self-reproducing Automata. University of Illinois Press Urbana, 1966. Burks A. (ed.)
3. *Wolfram S*. A new kind of science, Champaign, IL, Wolfram Media, 2002.
4. *Illachinski A*. Cellular Automata. A Discrete Universe. World Scientific Publishing, Singapoure. — 2001.
5. *Toffoli T., Margolus N.* Cellular Automata Machines, MIT Press, 1987.

6. *Chua L*. A nonlinear Dynamic Perspective of Wolfram's New Kind of Science, World Science, 2011.

7. *Adamatzki A.* Game of Life Cellular Automata, Springer-Verlag, 2010.

8. https://en.wikipedia.org/wiki/Cellular_automaton

9. *Chopard B., Droz M.* Cellular Automata Modeling of Physical Systems. Cambr. Univ. Press, 1998.

10. *Bandini S., Mauri G.* Multilayered cellular automata. Theoret Comput. Sci. volume 217, Issue 1. 1997. Pp.99-113.

11. *Goldengorin B., Makarenko A., Smilianec N*. Some applications and prospects of cellular automata in traffic problems Cellular Automata, Proceed. Int. Conf. ACRI'06, LNCS 4173, Springer, 2006. — P. 532–537

12. *Makarenko A., Krushinski D., Musienko A., Goldengorin B*. Towards Cellular Automata Football Models with Mentality Accounting. LNCS 6350. Springer – Verlag, 2010. — P. 149 – 153.

13. *Wonghhanasu Sartra* Cellular automata for pattern recognition. In: Emerging Applications of Cellular Automata. Ed. Alejandro Saliedo. InTech, Croatia, 2013. Pp. 53-68.

14. *Tsoutsouras V., Siracoulis G.Ch., Parlos G.P., Iliopoulos A.S.* Simulation of healthy and epilepiform brain activity using cellular automata. Int. J. of Bifurcation and Chaos. 2012. Volume 22, Issue 09. Pp. 1250229-52.

15. *Vlassopoulos N., Fates N., Berry H., Girau B.* An FPGA design for the stochastic Greenberg-Hastings cellular automata. Int. Conf. on High Performance Computing and Simulation – HPCS2010, Caen, France, 2010. Pp. 565-574.

16. *Didkivskiy A.M Makarenko O.S., Osaulenko V.M., Redchuk B.V.* Cellular automata concept – from micro to global levels. Proc. Of 1st WolfgangKummer Int. School on Astroparticle Quantum Gravity and Quantum Information, Aug. 3-16, 2015. Ed. S.S. Moscaliuk. Kyiv,: TIMPANI, 2016. PP. 75-84. (in Ukrainian)

17. *T'Hooft* The Cellular Autmation Interpretation of Quantum Mechanics. ArXiv: [qiant-ph]? 1405.1548. v3, 21 Dec. 2015. 259 p.

18. *Makarenko A.* Cellular automata methods in quantum Physics. Proc. Of 1st Wolfgang Kummer Int. School on Astroparticle Quantum Gravity and Quantum Information, Aug. 3-16, 2015. Ed. S.S. Moscaliuk. Kyiv,: TIMP.ANI, 2016. PP. 97-110. (in Ukrainian)

19. *Cooper S.* Barry Computability Theory. Chapman Hall/CRC (2003)

20. *Sipper M.* Co-evolving non-uniform cellular automata to perform computation. Physica D, Vol.92, Issue 3-4, 1996. Pp. 193-208.

21. *Fates N.* A Guided Tour of Asynchronous Cellular Automata. Cellular Automata and Discrete Complex Systems. LNCS 8155. J. Kari, M. Kutrib, Malcher A. (eds.), Springer, 2013. Pp. 15-30.

22. *Bandini S., Manconi S., Simone C.* Heterogeneous agents situated in heterogeneous space. Applied Artificial Intelligence. 2001. Volume 16, Issue 9-10. Pp. 831-852.

23. *Fates N.* Directed percolation phenomena in asynchronous elementary cellular automata. LNCS 4173, Springer-Verlag, 2006. Pp. 667-675.

24. https://www.nobelprize.org/nobel_prizes/physics/laureates/2016/press.html

Authors' Information

Lois Facchetti - *Mines Nancy – ADTEM, Nancy, France*

Alexander Makarenko - ***Institute for Applied System Analysis at National Tech. University of Ukraine (KPI) Kyiv, Ukraine*

e-mail: makalex51@gmail.com , makalex@i.com.ua