A very effective density classifier two-dimensional cellular automaton with memory

2009 J. Phys. A: Math. Theor. 42 485101

(http://iopscience.iop.org/1751-8121/42/48/485101)

IOP PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND THEORETICAL

J. Phys. A: Math. Theor. **42** (2009) 485101 (14pp) doi:10.1088/1751-8113/42/48/485101

A very effective density classifier two-dimensional cellular automaton with memory

Ramón Alonso-Sanz[1,2] **and Larry Bull**[1]

[1] Bristol Institute of Technology, University of the West of England, Frenchay Campus, Bristol BS16 1QY, UK
[2] ETSI Agronomos (Estadistica), Polytechnic University of Madrid, C.Universitaria. 28040, Madrid, Spain

E-mail: ramon.alonso@upm.es and Larry.Bull@uwe.ac.uk

Received 27 March 2009, in final form 14 September 2009
Published 11 November 2009
Online at stacks.iop.org/JPhysA/42/485101

Abstract
Conventional cellular automata (CA) are memoryless, i.e. the new state of a cell depends on the neighborhood configuration solely at the preceding time step. This paper considers an extension to the standard framework of CA by implementing memory capability in cells. It is shown that the HPP rule, one of the most important block automaton rules, endowed with the memory of the most frequent recent state, behaves as an excellent classifier of the density in the initial configuration, which surpasses the performance of the best two-dimensional density classifier reported in the literature.

PACS numbers: 05.50.+q, 45.00, 89.75.−k

(Some figures in this article are in colour only in the electronic version)

1. Cellular automata with memory

Cellular automata (CA) are discrete, spatially explicit extended dynamic systems. A CA system is composed of adjacent cells or sites arranged as a regular lattice, which evolves in discrete time steps. Each cell is characterized by an internal state whose value belongs to a finite set. The updating of these states is made simultaneously according to a common local transition rule involving only a neighborhood of each cell. Thus, if $\sigma_i^{(T)}$ is taken to denote the value of cell i at time step T, the site values evolve by iteration of the mapping:

$$\sigma_i^{(T+1)} = \phi\left(\left\{\sigma_j^{(T)}\right\} \in \mathcal{N}_i\right),$$

where ϕ is an arbitrary function which specifies the cellular automaton *rule* operating on the cells in the neighborhood ($\mathcal{N}$) of the cell i.

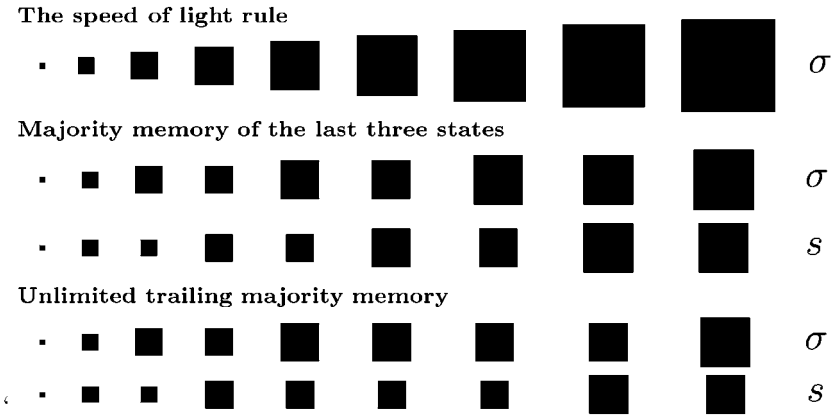

Figure 1. The *speed of light* cellular automaton. Memoryless formulation in the upper row of patterns, and two scenarios with majority memory below.

Conventional cellular automata are memoryless: i.e. the new state of a cell depends on the neighborhood configuration solely at the preceding time step. The standard framework of CA can be extended by implementing memory capabilities in cells:

$$\sigma_i^{(T+1)} = \phi\left(\left\{s_j^{(T)}\right\} \in \mathcal{N}_i\right),$$

with $s_j^{(T)}$ being a state function of the series of states of the cell j up to time-step T:

$$s_j^{(T)} = s\left(\sigma_j^{(1)}, \ldots, \sigma_j^{(T-1)}, \sigma_j^{(T)}\right).$$

Thus, in CA with memory, while the mappings ϕ remain unaltered, historic memory of all past iterations is retained by featuring each cell as a summary of its past states. In otherwords, cells *canalize* memory to the map ϕ.

A simple scenario will be taken for illustration purposes in figure 1: that of a two-dimensional automaton with two possible state values: $\sigma \in \{0, 1\}$, with the transition rule operating on nearest neighbors, in which a cell becomes (or remains) alive if any cell in its neighborhood is alive, but becomes (or remains) dead otherwise. The perturbation in figure 1 spreads as fast as possible, i.e. at the *speed of light*.

One simple memory implementation is that of keeping track of the most frequent of the last three states: $s_i^{(T)} = \text{mode}\left(\sigma_i^{(T-2)}, \sigma_i^{(T-1)}, \sigma_i^{(T)}\right)$, with initial conditions $s_i^{(1)} = \sigma_i^{(1)}$, $s_i^{(2)} = \sigma_i^{(2)}$. Such majority memory, referred to as $\tau = 3$ majority memory, exerts a characteristic inertial effect that tends to restrain the CA dynamics, as shown in figure 1 regarding the spread from a single seed. Increasing the length of memory increases the inertial effect, as is shown in figure 1 with unlimited trailing memory $s_i^{(T)} = \text{mode}\left(\sigma_i^{(1)}, \ldots, \sigma_i^{(T)}\right)$, with $s_i^{(T)} = \sigma_i^{(T)}$ in the case of a tie: $\text{card}\{1\} = \text{card}\{0\}$. With memory limited to the last three states, the patterns in figure 1 remain stable for two time steps; thus, the speed is halved, whereas with unlimited trailing memory the dynamics might be described in terms of *punctuated equilibrium*, i.e. periods of quiescence altered by changes that take place at well-defined steps (the punctuation marks). Shown in figure 1 are the powers of two time steps when the automaton fires a new perimeter of live cells, so that the duration of the stable periods tends to increase with T.

The memory mechanism considered here differs from that of other CA with memory reported in the literature, in particular with that referred to as *higher order* (in-time) CA,

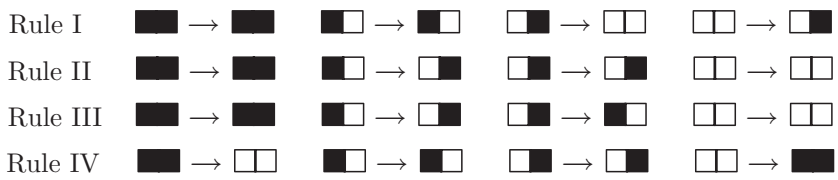

Figure 2. One-dimensional block cellular automata.

such as the second-order reversible formulation $\sigma_i^{(T+1)} = \phi\left(\left\{\sigma_j^{(T)}\right\} \in \mathcal{N}_i\right) \ominus \sigma_i^{(T-1)}$, which remains reversible in the form $\sigma_i^{(T+1)} = \phi\left(\left\{s_j^{(T)}\right\} \in \mathcal{N}_i\right) \ominus \sigma_i^{(T-1)}$ [3]. Other possible memory implementations, not considered here, are (i) to refer a cell of the neighborhood *delayed* a number of time steps in the past. So, for example, in the context of elementary CA the central cell may be referenced not at T but at $T-1$: $\sigma_i^{(T+1)} = \phi\left(\sigma_{i-1}^{(T)}, \sigma_i^{(T-1)}, \sigma_{i+1}^{(T)}\right)$, (ii) the transition rule may be *extended* to *explicitly* consider the influence at time $T-1$, e.g. $\sigma_i^{(T+1)} = \psi\left(\sigma_i^{(T-1)}, \sigma_{i-1}^{(T)}, \sigma_i^{(T)}, \sigma_{i+1}^{(T)}\right)$ or (iii) memory can be operative only in the cell to be updated: $\sigma_i^{(T+1)} = \phi\left(\sigma_{i-1}^{(T)}, s_i^{(T)}, \sigma_{i+1}^{(T)}\right)$ [22].

To the best of our knowledge the study of the effect of memory on CA has not been given the attention that it deserves. Thus, as two symptomatic examples, Wuensche and Lesser ([26], p 15) just mention the possibility of *historical time reference*, which is excluded from their general study as 'it would result in a qualitatively different behavior', whereas Wolfram [24], in the context of higher order linear CA, refers to a 'somewhat involved analysis not performed here'.

We have studied the effect of memory embedded in cells in several CA [2] and Boolean network [4, 5] scenarios. In a recent study, we analyzed the influence of memory in the spatialized prisoner's dilemma [1].

2. Block cellular automata

A partitioned (or block) CA is a CA with a partitioning scheme such that the set of cells are partitioned in some periodic way: every cell belongs to exactly one block, and any two blocks are connected by a lattice translation. The update rule of a partitioned CA takes as input an entire block of cells and outputs the updated state of the entire block. The rule is then applied alternatingly to the even and to the odd translations.

In the so-called Margolus neighborhood [21], blocks are formed by 2×2 squares of cells in two-dimensional lattices, or simply couples of adjacent cells in one-dimensional registers. Four block cellular automata rules in the one-dimensional context are given in figure 2.

The left column of patterns of figure 12 shows the spatio-temporal patterns of these elementary one-dimensional block CA starting from a single full block, and starting at random. The registers in the figure are of size $N = 60$, an even number size that can be perfectly partitioned in blocks of size 2. Incidentally, in registers with odd number size, such partition is not feasible, so that one border cell remains 'untreated' at every updating. This might help to explain the worse efficiencies achieved in the density classification task in odd-size registers reported in the next section. The evolution in the memoryless model of the two initially adjacent cells under rule III in figure 12 may be interpreted as of a form of particles collision. In fact rule III is the one-dimensional simple version of the so-called HPP

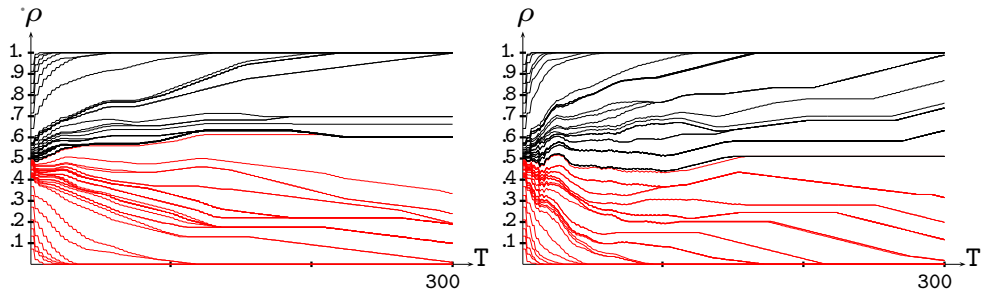

Figure 3. Rules II (left) and III (right) densities with $\tau = 3$ majority memory.

rule [14, 15], a seminal rule in gas computer modeling, properly stated in the 2D scenario in figure 10.

Figure 12 also shows the effect of the embedding memory of the last three states and that of unlimited trailing memory.

Rules I, III and IV are reversible, and rules II and III conserve the number of active sites [8], i.e. they keep unaltered the initial density. When implementing the majority memory, these rules lose the number conserving property [10], so that the evolution of the density of active cells (ρ_t) is that shown in figure 3.

3. Density classification task

The density classification task (DCT), also referred to as the majority problem, is the problem of finding cellular automaton rules that accurately determine if an initial configuration (IC) has more or less 1s than 0s, i.e. if the initial density (ρ_0) is over or under 0.5. Ideally a correct solution of the DCT must eventually set all cells to zero if $\rho_0 < 0.5$, and must eventually set all cells to one if $\rho_0 > 0.5$. The desired eventual state is unspecified if $\rho_0 = 0.5$.

While solving the DCT is a trivial task for any computational system with central control, this is not the case for fully distributed systems, with local processing, as cellular automata. Solving the DCT has attracted attention in the specialized literature, due to the hope that good results achieved by a specific technique for DCT may be useful regarding other CA-based problems. This problem is an important test case in measuring the computational power of cellular automaton systems.

Although the DCT was proposed in the seventies, only in 1994 was the solution to the problem, as formulated, proven to be impossible [17]. Thus, research efforts shifted to looking for the best rule dealing with the DCT, i.e. the rule for which the fraction of the possible starting configurations that are correctly classified is the highest.

The DCT may be solved if one relaxes the definition by which the automaton is said to have recognized the initial density [9, 11]. But we will keep here the unaltered specified *natural* criterion: $\rho_* = 0$ if $\rho_0 < 0.5$, $\rho_* = 1$ if $\rho_0 > 0.5$, denoting as ρ_* the final density achieved.

4. One dimension

The dynamics of the density under rules II and III with memory in figure 3 shows as an overall rule that the patterns tend to drift to fixed configurations either of all 1s or of all 0s depending

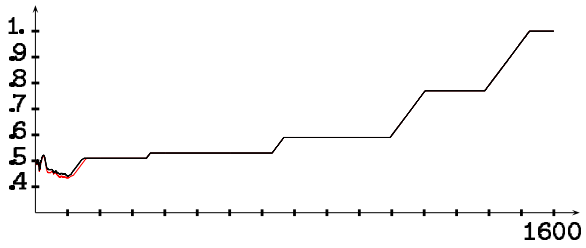

Figure 4. Further evolution of the instances that appear stable in the right frame of figure 3.

upon whether within the initial configuration $\rho_0 > 0.5$ or the contrary. So far, it is foreseeable that rules II and III endowed with memory in cells of the most frequent of the last three states, i.e. $\tau = 3$ majority memory, will produce good results in classifying density by leading to steady configurations that readily indicate the correct classification. It becomes apparent from figure 3 that these rules with memory very soon relax to the correct fixed point if the initial density is either $\rho \geqslant 0.6$ or $\rho \leqslant 0.4$.

Rule II is equivalent to the elementary CA rule 184 [9], whose properties regarding density classification with memory were outlined in [6, 7]; thus, we will focus here on rule III. Two examples of this rule with $\tau = 3$ majority memory starting from low and high initial densities are given in figure 13, showing how this rule with memory readily relaxes in both cases to the fixed point that correctly classifies the initial configuration.

The drift of density is much slower in the oversampled [0.4, 0.6] interval in figure 3, but the steady configurations are ever reached. This is so even for the particular cases that appear stabilized up to $T = 300$ in the right frame of figure 3, corresponding to initial actual densities 0.4975 and 0.5025 but coincident after $T = 150$, whose further evolution is shown in figure 4. Thus, only the configuration with initial density 0.4975 in figure 4 is misclassified by the $\tau = 3$ rule III. The spatio-temporal pattern starting from the correctly classified $\rho_0 = 0.5025$ is shown in figure 14, in which the goal of every cell at 1 is achieved unusually late, at $T = 1530$.

In a simulation of 10 series of 1000 uniformly distributed initial densities in the [0.0, 1.0] interval, with every simulation run up to $T = 1600$ in a register with $n = 400$ cells, only 66 densities in the [0, 48, 0.52] interval were incorrectly classified. The GKL rule[3] [12, 13], one of best density classifier rules, when applied to the same series misclassified 160 densities.

Both GKL and rules with $\tau = 3$ majority memory have in common the presence of the majority operation, that seems to be in the origin of excellent results in the density task. In fact the majority rule has been implemented regarding the DCT not only in the perfectly structured CA context, but also in the so-called CA on graphs, e.g., in the *small-world* networks in [23], and in the noisy communication between units and asynchronous updating contexts addressed in [18]. This study adopts the majority rule (still with radius 3) as a better alternative to the GKL, which shows a poor DCT performance in the aforementioned non-idealized structure of interactions. The authors support their choice of the majority rule in [18] as 'a plausible heuristic to reaching the consensus' because 'it is reasonable to hypothesize that in real-world systems the units make their decisions by using simple heuristics that are robust against errors and do not depend on the precise structure of interactions'.

[3] $$\sigma_i^{(T+1)} = \begin{cases} \text{mode}\left(\sigma_i^{(T)}, \sigma_{i-1}^{(T)}, \sigma_{i-3}^{(T)}\right) & \text{if} \quad \sigma_i^{(T)} = 0 \\ \text{mode}\left(\sigma_i^{(T)}, \sigma_{i+1}^{(T)}, \sigma_{i+3}^{(T)}\right) & \text{if} \quad \sigma_i^{(T)} = 1 \end{cases}$$

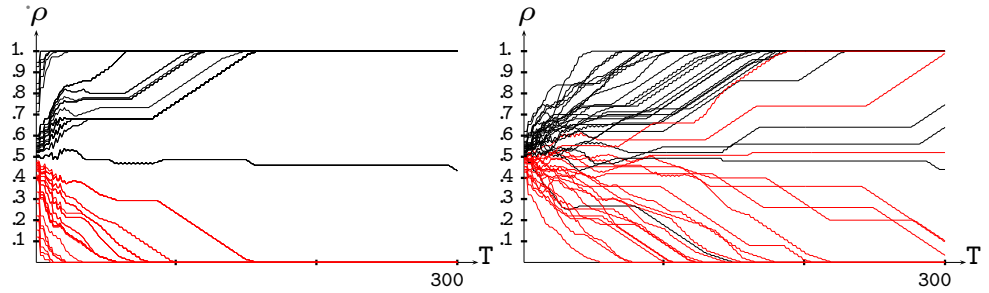

Figure 5. Evolution of the density in rule III with $\tau = 3$ majority memory in a $n = 150$ register. Left: uniform simulation of initial density. Right: binomial simulation of initial cell states.

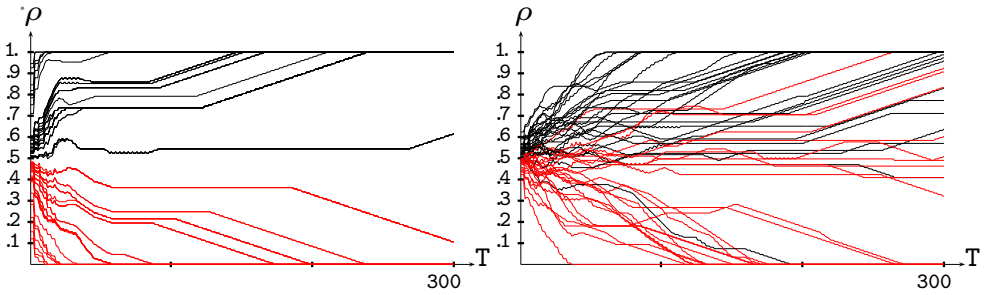

Figure 6. Density evolution in the scenarios of figure 5, but in a $n = 149$ register.

To test the performance of rules in more challenging scenarios, initial configurations are to be generated *binomially* distributed, i.e. with every cell in the register given a state value equiproblably. In 10^4 binomially generated IC, discarding the 419 configurations in which the number of zeros is exactly equal to that of ones, i.e. the unclassificable $\rho_0 = 0.5$, resulted an efficiency in the correct classification (referred to as ϵ_τ in simulations with majority memory of length τ) of $\epsilon_3 = 81.002\%$. Increasing the length of the trailing memory up to $\tau = 4$ seems to slightly increase the capacity to discriminate density, $\epsilon_4 = 81.192\%$ in our simulation. Beyond this trailing length, inertial effects characteristic of the majority memory, oppose the drift to the all 1s or all 0s intended in the density classification: $\epsilon_5 = 78.061\%$, $\epsilon_6 = 78.238\%$, $\epsilon_7 = 75.952\%$, $\epsilon_8 = 75.775\%$. It seems that $\tau = 3$ as length of the majority memory is a good (and simple) choice regarding the density task.

In order to compare the block rules with memory here with other rules effectively solving the density task, the remaining result reported in this paper concerning the one-dimensional scenario apply for registers of size $n = 149$ and $n = 150$. Thus, in figure 5, with $n = 150$ the $\tau = 3$ rule III proves to be faster in its drift to a steady configuration compared to the $n = 400$ scenario, mainly indicating the correct density classification. Thus, only one instance is misclassified in the left frame of figure 5, with $\rho_0 = 0.5067$, whereas in the binomially simulated IC in the right frame of figure 5, only three instances are incorrectly classified, one of which is shown in figure 15. Figure 6 refers to the odd size $n = 149$. No instance is misclassified in the left frame of figure 6, and only five instances are incorrectly classified in the binomially IC simulated right frame.

Table 1. Percentage of correctly classified densities in a simulation of one million of binomially generated initial configurations. Rule III with majority memory of length τ in one-dimensional registers of size n.

	$\tau = 3$	$\tau = 4$	$\tau = 5$	$\tau = 6$
$n = 150$	85.049	85.641	82.530	82.839
$n = 149$	81.562	82.026	79.152	79.346

The results on efficiency in a simulation of one million of binomially generated initial configurations in registers of sizes 150 and 149 (similar sizes to that in [25]), evolving up to $T = 2000$, are shown in table 1. In the $n = 150$ context of table 1, 65 215 $\simeq 6.5\%$ IC were unclassificable ones with $\rho_0 = 0.5$.

The figures in table 1 are notable compared to those reported in the literature; nevertheless, they are surpassed for the best ones, those reported in [25], which reach $\epsilon_3 = 88.9\%$. It turns out that the efficiency in the simulations in registers of even size is higher than in those in the $n = 149$ size registers (in which the GKL rule reaches $\epsilon_3 = 81.6\%$), and that $\tau = 4$ is a slightly better choice than $\tau = 3$. As noted above, increasing the trailing length of memory increases the inertial effects that oppose the density discrimination, so that the figures of efficiency in table 1 are lower for $\tau = 5, 6$.

An example of misclassification with $n = 149$ is given in figure 16.

A low fraction of the initial configurations turn out to be unclassified due to the lack of convergence to any of the extreme configurations, i.e. all 1s or all 0s. These unclassified configurations are counted as *misclassified* to compute the efficiency in the density discrimination. An example is given in figure 17, in which case the density gets fixed to 0.5 as soon as the pattern becomes perfectly symmetric. This is usually the kind of dynamics that impedes the density discrimination, which explains why unclassified configurations are mostly found dealing with even-size registers. In the $n = 150$ context of table 1, the number of unclassified configurations were 6950 $\simeq 0.7\%$, 4218, 6625 and 4322 from $\tau = 3$ up to $\tau = 6$. With $n = 149$, the number of unclassified configurations were lower: 1756, 1130, 0 and 3. Nevertheless, misclassified IC, the important numerical component of the not correctly classified configurations, are notably less frequent in the even-size lattice simulations.

5. Two dimensions

Figure 7 refers to two-dimensional lattices [20] evolving under the HPP rule (as defined in figure 10) with $\tau = 3$ majority memory. The classification under this rule becomes remarkably straightforward, moreover if attention is paid to the fact that evolution is shown up to $T = 100$, not up to $T = 300$, as in the equivalent figure in the one-dimensional scenario, i.e. figure 5. Figure 8 shows two examples of the evolution in the first four time steps, starting from high ($\simeq$90%) and low ($\simeq$10%) densities. As soon as memory becomes operative, i.e. at $T = 4$, the patterns are dramatically skewed to follow their correct classifier steady configuration.

The results on efficiency in a simulation of 10^5 binomially generated initial configurations in 22×22, and 21×21 size lattices (similar sizes to that in [25]), as well as in the bigger lattices of sizes 32×22, and 31×31, are shown in table 2. The automata in table 2 evolve up to $T = 200$ in $n = 22$ lattices, thus one tenth of the time steps of that in table 1, up to $T = 300$ in $n = 21$ lattices and up to $T = 500$ in $n = 32, 31$ side size lattices. In the $n = 22$ and $n = 32$ contexts of table 2, 3.6 and 2.3% configurations respectively were

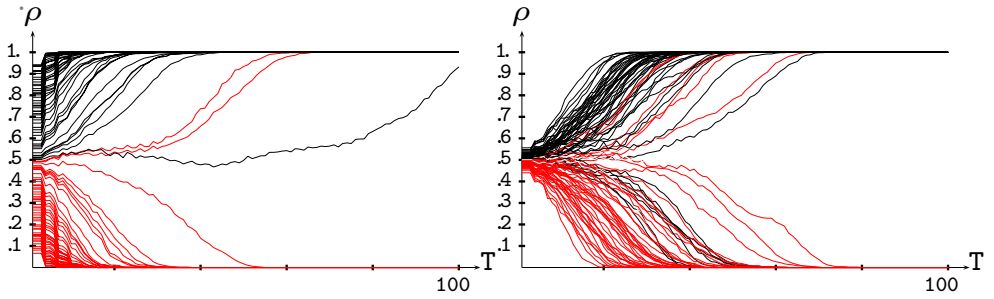

Figure 7. Evolution of the density in the HPP rule with $\tau = 3$ majority memory in a 22×22 lattice. Left: uniform simulation of initial density. Right: binomial simulation of initial cell states.

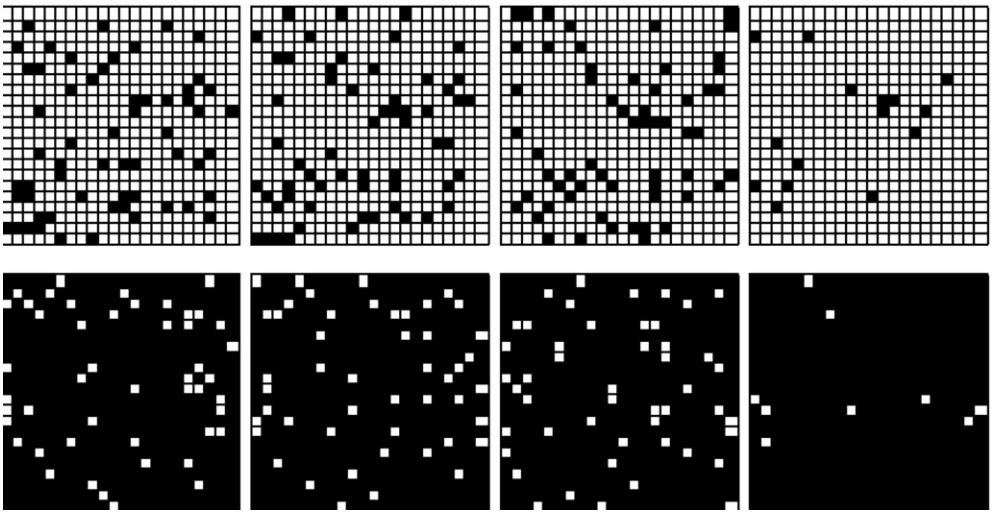

Figure 8. Patterns up to $T = 4$ in the HPP rule with $\tau = 3$ majority memory in a 22×22 lattice.

unclassificable with $\rho_0 = 0.5$. The number of IC unclassified due to the lack of convergence to any of the extreme configurations is much lower in the two-dimensional context compared to the one-dimensional, under the 0.1%. The figures for the small sizes in table 2 are 3, 372, 11 and 97 for $n = 22$ (from $\tau = 3$ up to $\tau = 6$), and 40, 22, 316 and 142 for $n = 21$. The unclassified 2D initial configurations often become configurations close to the extreme steady configurations as those in figure 9, generated from $\rho_0 = 0.523$ and $\rho_0 = 0.496$, in the $\tau = 4$, $n = 22$ scenario.

Let us stress here that the convergence to the steady configurations is very fast. This is so even also when dealing with binomially generated IC, as already seen graphically with $\tau = 3$ in figure 7, and quantified in table 2. The mean values of the time up to convergence to steady configurations of classified IC are notably higher in odd-side size lattices than in their close even-size ones. Thus, in the odd-side size lattices not only is the efficiency lower than in the even-size, but the convergence to the steady configurations takes longer, approximately double the number of time steps.

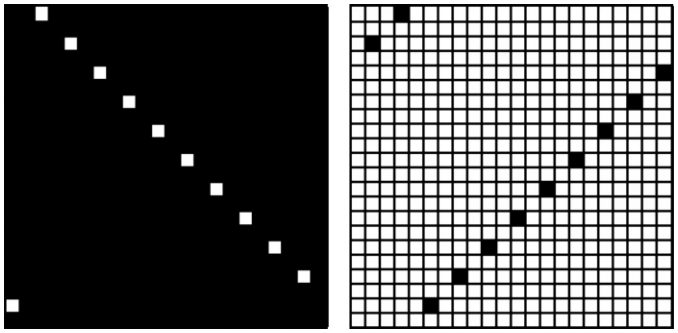

Figure 9. Stable configurations in the HPP rule with $\tau = 4$ majority memory in a 22×22 lattice.

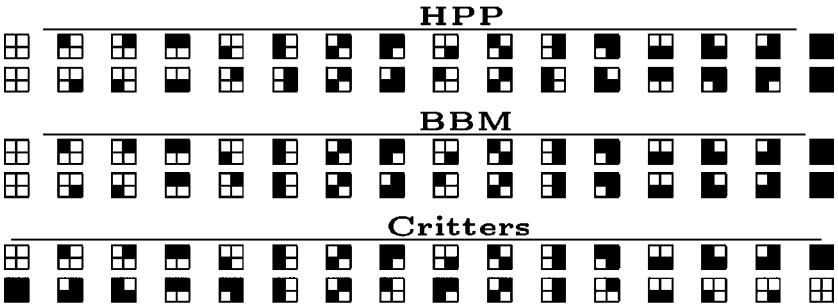

Figure 10. Two-dimensional block cellular automata. The lower row shows the evolution of the upper row by rule.

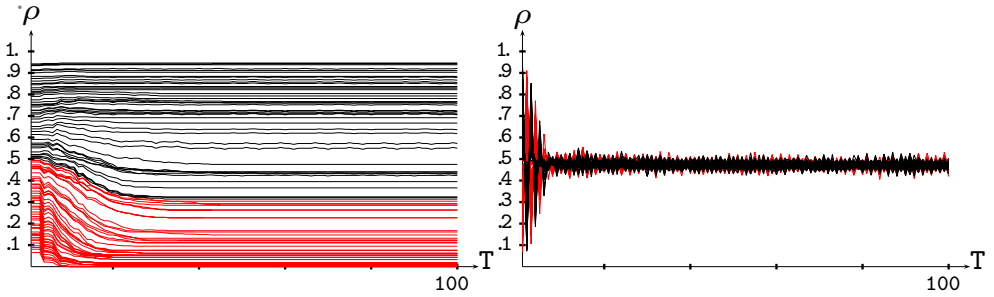

Figure 11. Evolution of the density in the BBM (left) and Critters (right) rules with $\tau = 3$ majority memory in a 22×22 lattice.

Table 2. Percentage of correctly classified densities and average time up to convergence of classified configurations. HPP rule with majority memory of length τ in two-dimensional lattices of size $n \times n$. Simulation of 10^5 binomially generated initial configurations.

	$\tau = 3$	$\tau = 4$	$\tau = 5$	$\tau = 6$
$n = 22$	88.648 43	92.767 42	87.368 52	89.779 48
$n = 21$	84.235 83	87.877 84	82.821 99	84.811 96
$n = 32$	87.171 66	91.131 63	85.798 79	87.744 74
$n = 31$	82.646 130	85.076 130	81.157 157	83.811 152

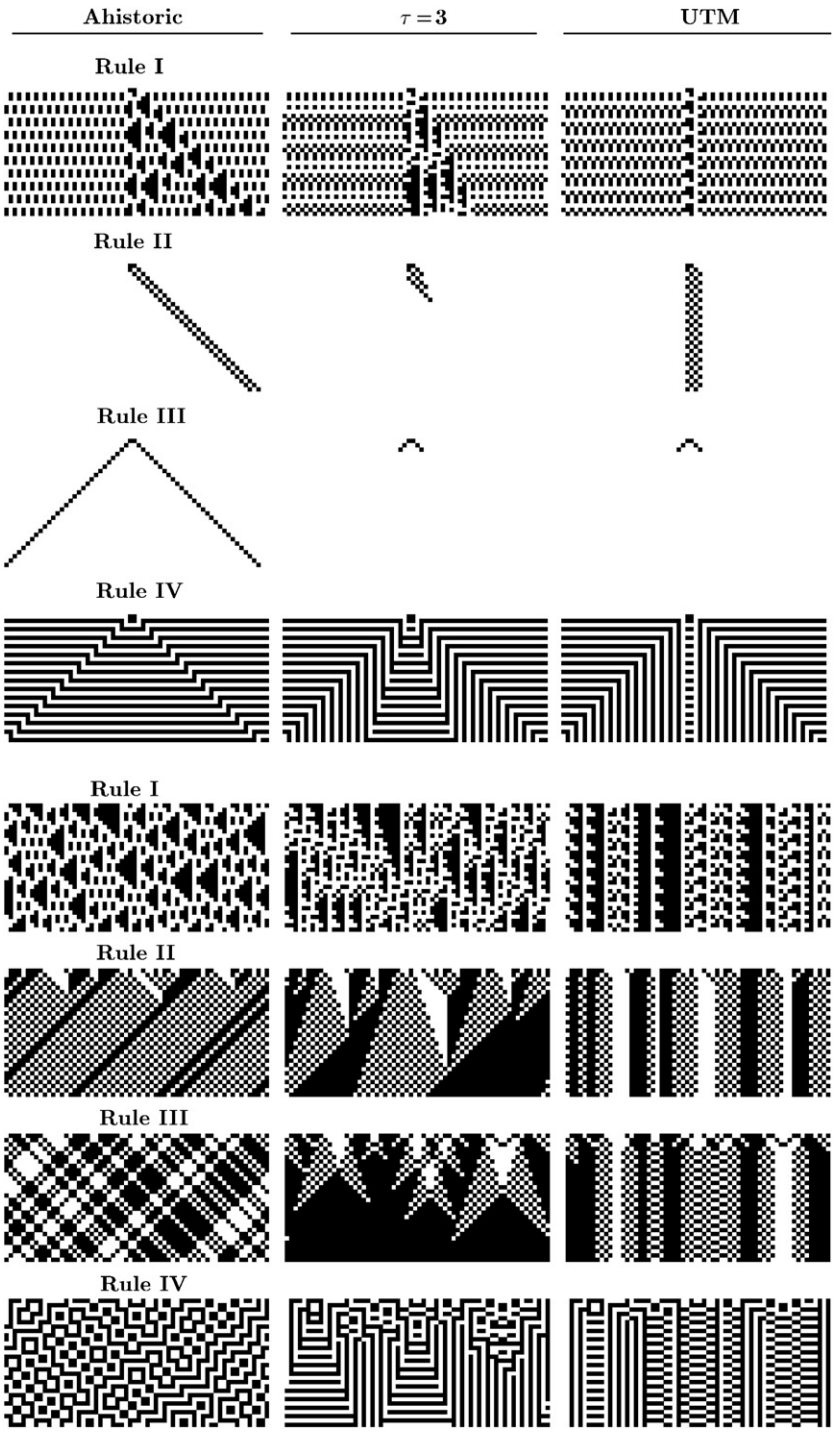

Figure 12. Elementary one-dimensional block CA starting from a single full block (upper), and at random (below).

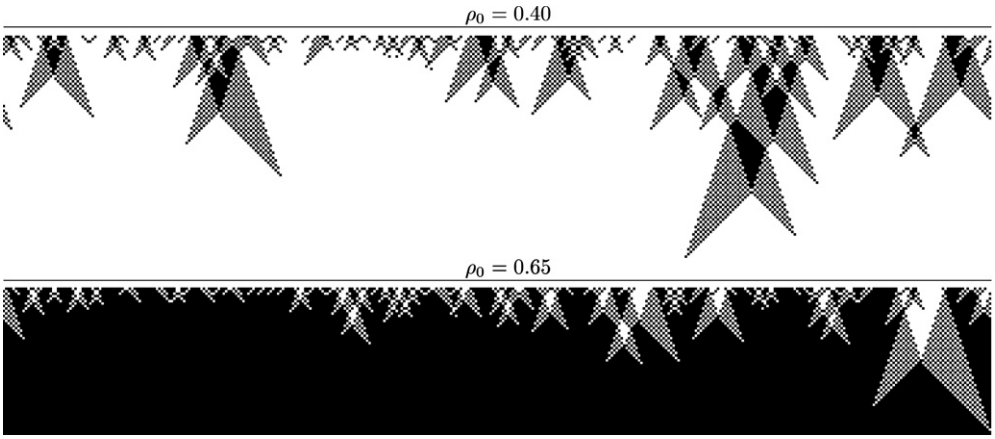

Figure 13. Rule III block CA with $\tau = 3$ majority memory, starting at random.

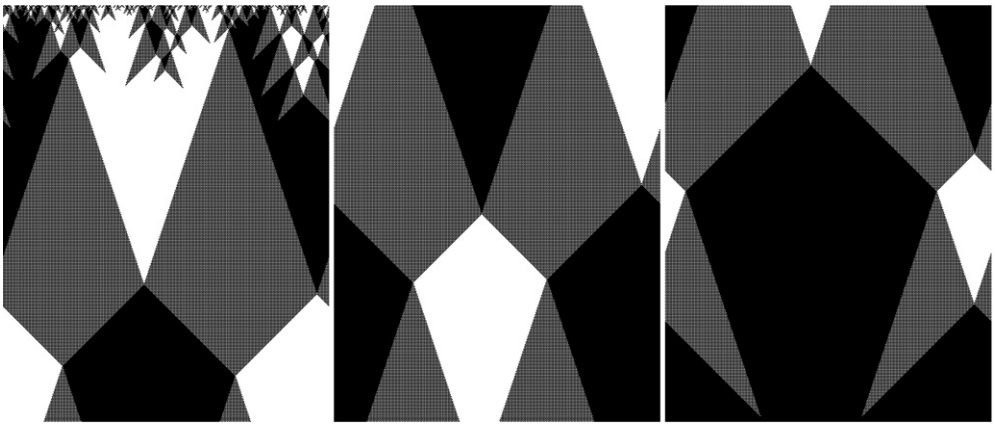

Figure 14. Spatio-temporal patterns under rule III with $\tau = 3$ majority memory from a configuration with $\rho = 0.5025$. Evolution up to $T = 1530$.

As the size of the lattices increases from 21–22 to 31–32, the performance of the rules decreases, and the convergence takes longer. In this comparison, the difference in the convergence time seems notable, but the decay in the classification performance is not too high, lower than 2%.

The efficiencies in the $n = 21$ side lattice table 2 overcomes that of the best rule reported in [19]: $\epsilon \simeq 70.0\%$, and that corresponding to $\tau = 4$ also surpasses the highest efficiency reported in [25]: $\epsilon = 83.27\%$. In the odd-side size lattice, the one-dimensional rule III has been applied to the two odd border sides at every time step. Without this updating of border cells, the efficiency in the density discrimination declines dramatically. The progress made by the HPP rule with memory in the even-side size lattice, in which case the block partitioning covers the whole lattice every time step, is highly remarkable.

The maximum efficiency in table 2 is reached, as in the one-dimensional context, when tracking memory of the four last time steps. Further comparison to other DCT rules, particularly those reported in [16], will be made in the next section.

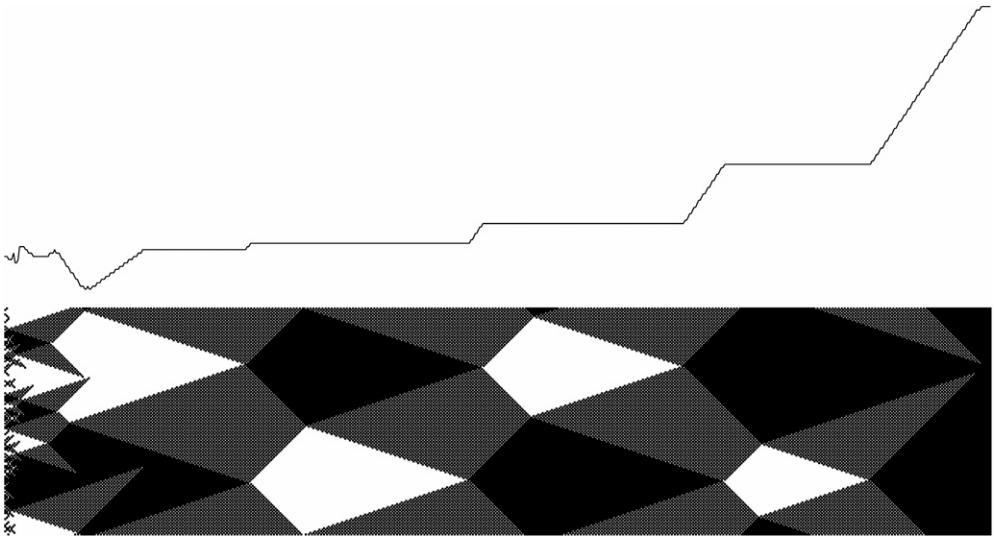

Figure 15. Wrong classified density $\rho_0 = 0.4933$. Evolution up $T = 650$ in a register of size $n = 150$. The density evolution is shown over the spatio-temporal pattern.

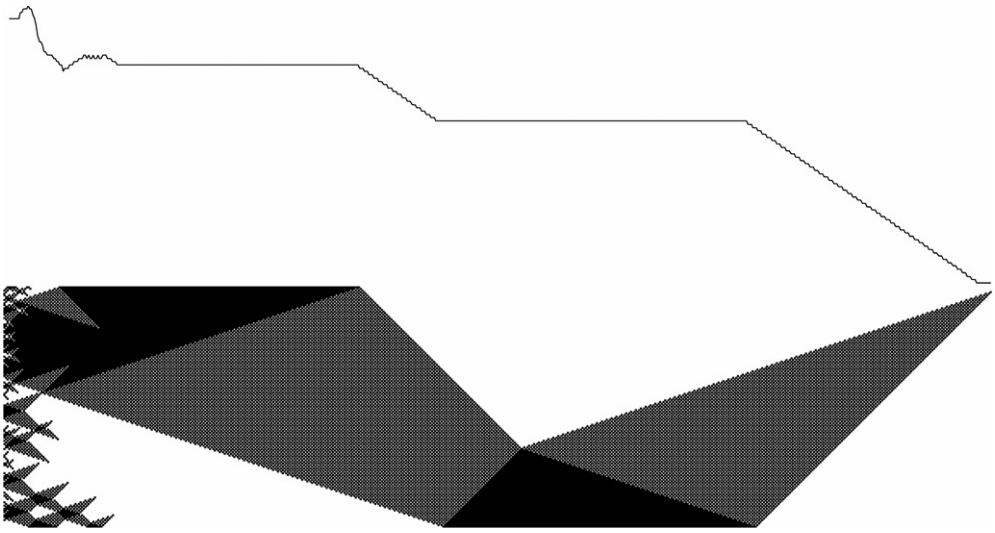

Figure 16. Wrong classified density $\rho_0 = 0.5370$. Evolution up $T = 600$ in a register of size $n = 149$.

Other 2D block cellular automata have been checked regarding the DCT, but showed poor results. Thus for example, in figure 11, the Billiard Ball Machine (BBM) [21], seems not to be active enough to promote the drift to the extreme steady configurations, whereas the *Critters* rule (a rule supporting *gliders* [21]) leads to configurations oscillating nearly 0.5, regardless the initial density. Both BBM and *Critters* rules are defined in figure 10. Evolutionary techniques [16] may help in the search for more efficient, albeit also more sophisticated, partitioned CA density classifiers.

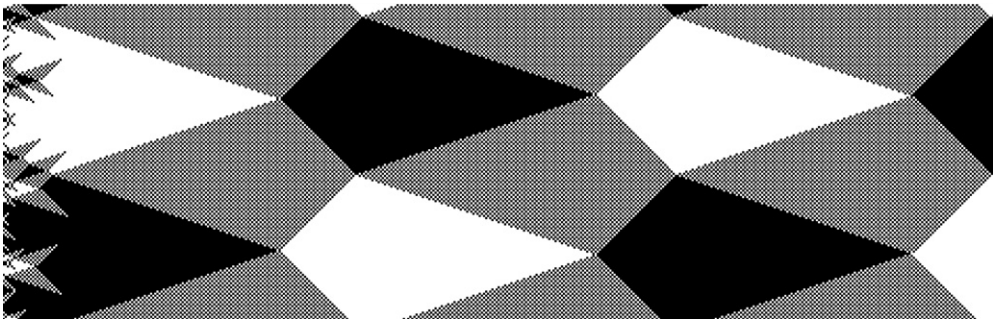

Figure 17. Unclassified configuration. Evolution up to $T = 650$, $n = 149$.

6. Conclusion

When the cells of a simple block cellular automaton, well known in the literature as HPP, are endowed with majority memory of their last states, the HPP automaton proved to be a remarkably good density classifier when operating in two-dimensional lattices. The percentage of correct classification in the best memory implementations reaches 90% (some over this figure in even-side size lattices, and close to it in odd-side size lattices), surpassing the performance of the best two-dimensional density classifier reported in the literature. In the one-dimensional scenario, the HPP automaton with memory turns out to also be a notable density classifier, though other conventional rules of radius 3 are known to outperform it.

Although no theoretical attempt is made in this experimental study to explain the good performance of the HPP rule endowed with majority memory in cells, several heuristic reasons may be argued to support it: (i) HPP is a number conserving rule, as the reputed DCT-solver elementary rule 184, (ii) HPP allows for information transmission as the good classifiers reported in [16] in 2D lattices, and, last but not least, (iii) the majority operation seems to be present in most of the good density classifiers, such as the GKL and the non-local majority in structured CA [16], and other generalized majority rules in non-local networks [23] and in noisy environments [18].

7. Acknowledgment

This work was supported under EPSRC grant no EP/E049281/1.

References

[1] Alonso-Sanz R 2009 Spatial order prevails over memory in boosting cooperation in the iterated prisoner's dilemma *Chaos* **19** 023102
[2] Alonso-Sanz R 2009 Cellular automata with memory *Encyclopedia of Complexity and Systems Science* ed R Meyers (New York: Springer) pp 823–48 and the references therein
[3] Alonso-Sanz R 2003 Reversible cellular automata with memory *Physica* D **175** 1–30
[4] Alonso-Sanz R and Cardenas J P 2008 On the effect of memory in one-dimensional $K = 4$ automata on networks *Physica* D **237** 3099–108
[5] Alonso-Sanz R and Bull L 2008 Boolean networks with memory *Int. J. Bifurcation Chaos* **18** 3789–97
[6] Alonso-Sanz R and Bull L 2010 One-dimensional coupled cellular automata with memory: initial investigations *J. Cellular Automata* at press
[7] Alonso-Sanz R and Bull L 2008 Elementary coupled cellular automata with memory *AUTOMATA-2008* ed Adamatzky *et al* (Bristol: Luniver) pp 72–99

[8] Boccara N and Fuks H 1998 Cellular automata rules conserving the number of active sites *J. Phys. A: Math. Gen.* **31** 6007–18
[9] Capcarrere M S, Sipper M and Tomassini M 1996 Two-state, $r = 1$ cellular automaton that classifies density *Phys. Rev. Lett.* **77** 4969
[10] Fuks H and Sullivan K 2007 Enumeration of number-conserving cellular automata rules with two inputs *J. Cellular Automata* **2** 141–8
[11] Fuks H 1997 Solution of the density classification problem with two cellular automata rules *Phys. Rev.* E **55** R2081–4
[12] Gacs P, Kurdyumov G L and Levin L A 1978 One-dimensional uniform arrays that wash out finite islands *Problemy Pederachi Infomatsii* **14** 92–8
[13] Gonzaga de Sá P and Maes C 1992 The Gacs–Kurdyumov–Levin automaton revisited *J. Stat. Phys.* **67** 507–22
[14] Hardy J, Pazzis O and Pomeau Y 1976 Molecular dynamics of a classical lattice gas: transport properties and time correlation functions *Phys. Rev.* A **13** 1949–61
[15] Hardy J, Pomeau Y and Pazzis O 1973 Time evolution of a two-dimensional model system. I. Invariant states and time correlation functions *J. Math. Phys.* **14** 1746–59
[16] Jimenez-Morales F, Crutchfield J P and Mitchell M 2001 Evolving two-dimensional cellular automata to perform density classification: a report on work in progress *Parallel Comput.* **27** 571–85
[17] Land M and Belew R 1995 No perfect two-state cellular automata for density classification exists *Phys. Rev. Lett.* **74** 25
[18] Moreira A A, Mathur A, Diermeier D and Amaral L A N 2004 Efficient system-wide coordination in noisy environments *Proc. Natl Acad. Sci. USA* **101** 12085–90
[19] Oliveira G M B and Siqueira S R C 2006 Parameter characterization of two-dimensional cellular automaton rule space *Physica* D **217** 1–6
[20] Reynaga R and Amthauer E 2003 Two-dimensional cellular automata of radius one for density classification task $\rho = 1/2$ *Pattern Recognit. Lett.* **24** 2849–56
[21] Toffoli T and Margolus N 1987 *Cellular Automata Machines* (Cambridge, MA: MIT Press)
[22] Rohlf T and Tsallis C 2007 Long-range memory elementary 1D cellular automata: dynamics and nonextensivity *Physica* A **379** 465–70
[23] Watts D J 1999 *Small Worlds* (Princeton, NJ: Princeton University Press)
[24] Wolfram S 1984 Algebraic properties of cellular automata *Physica* D **10** 1–35
[25] Wolz D and Oliveira P P B 2008 Very effective evolution techniques for searching cellular automata spaces *J. Cellular Automata* **3** 289–312
[26] Wuensche A and Lesser M 1992 *The Global Dynamics of Cellular Automata* (Reading, MA: Addison-Wesley)